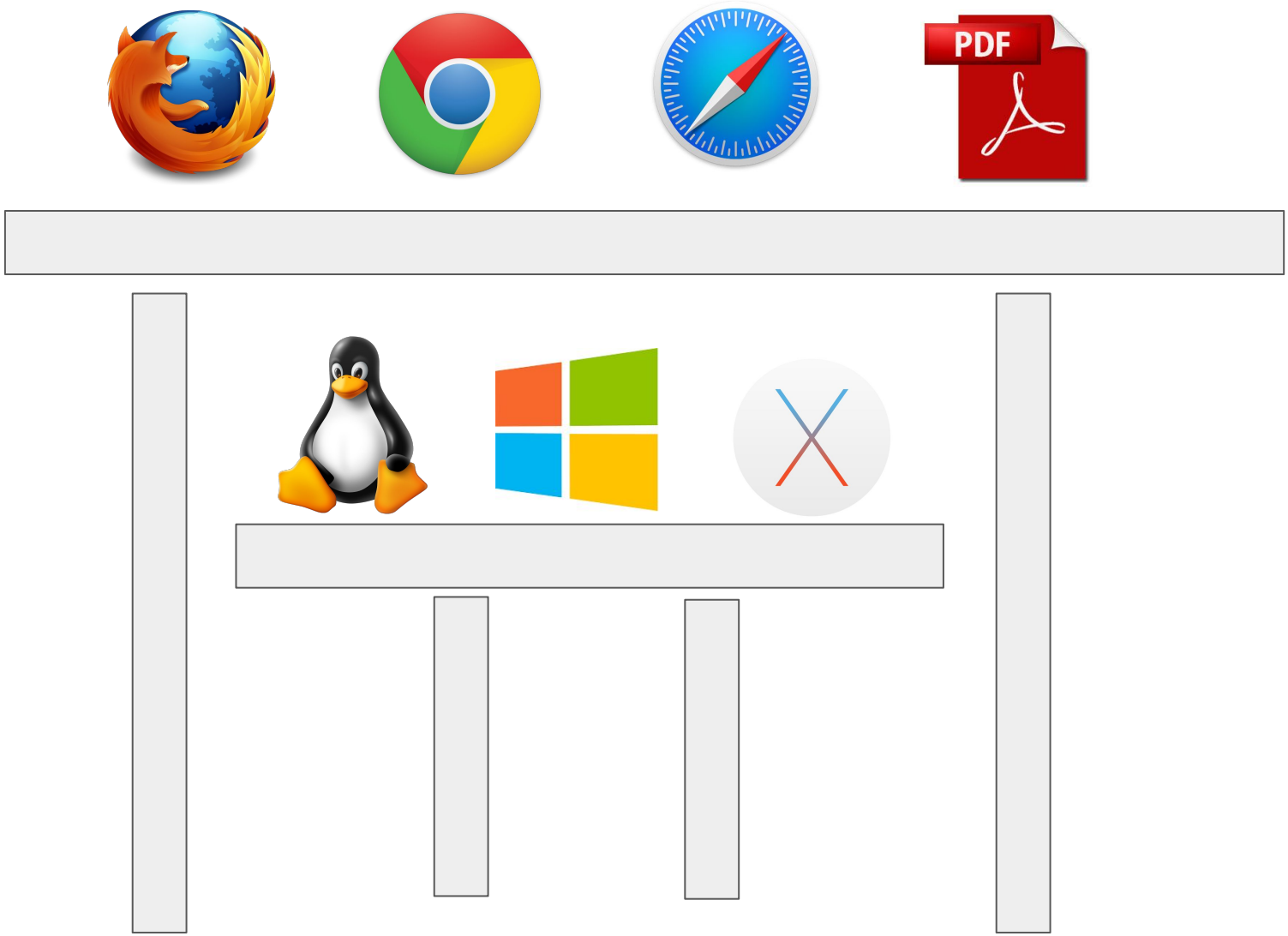
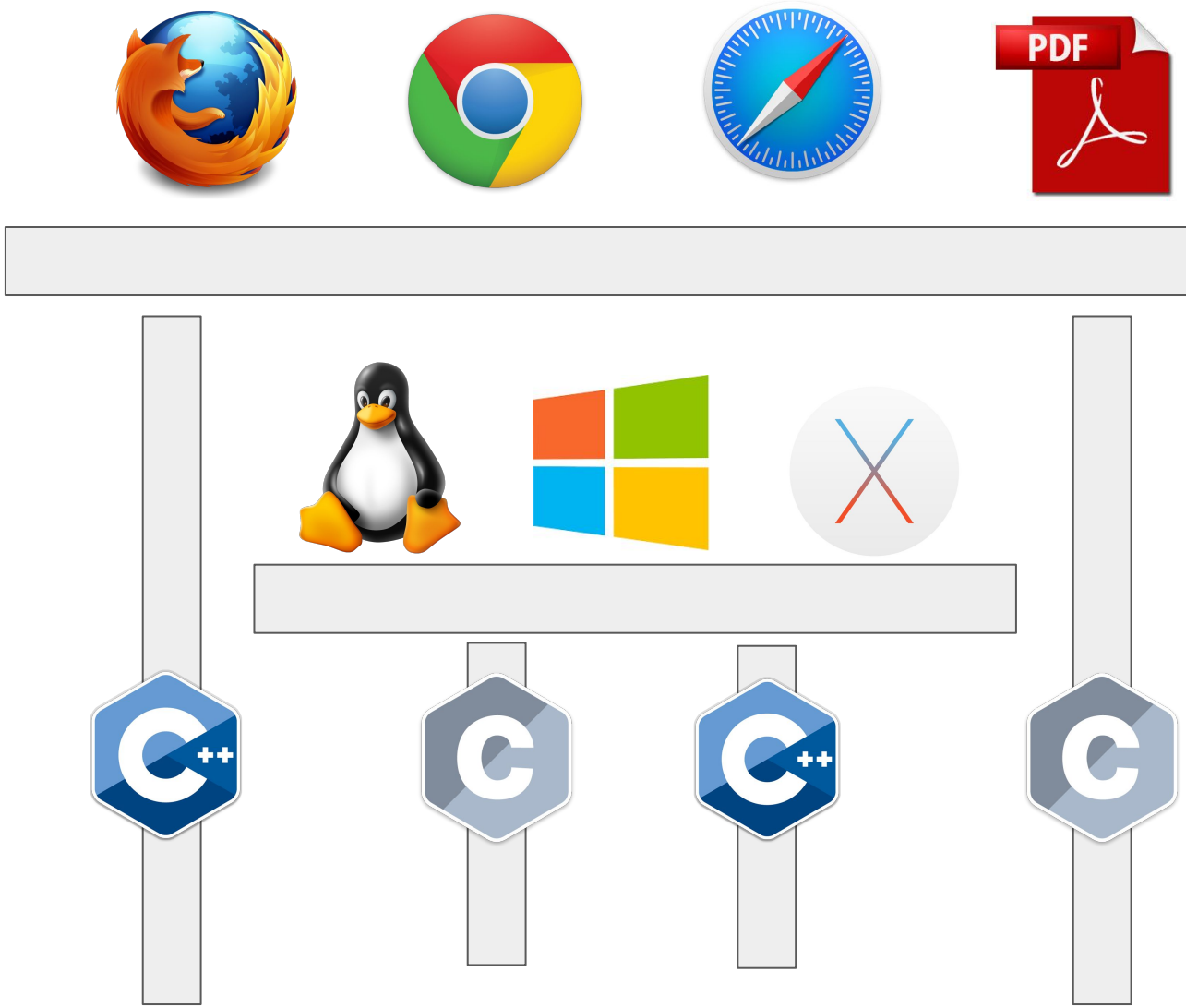
Feel me Flow: A Review of Control-Flow Integrity Methods for User and Kernel Space

Irene Díez-Franco, Igor Santos
DeustoTech, University of Deusto
irene.diez@deusto.es isantos@deusto.es

Rationale

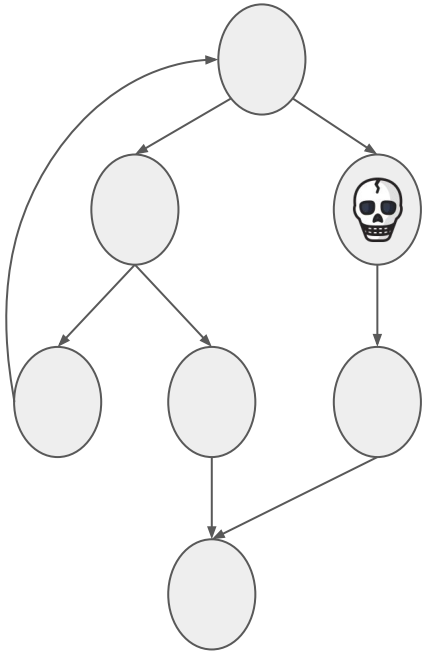


Rationale



Code injection attacks

Code injection attacks

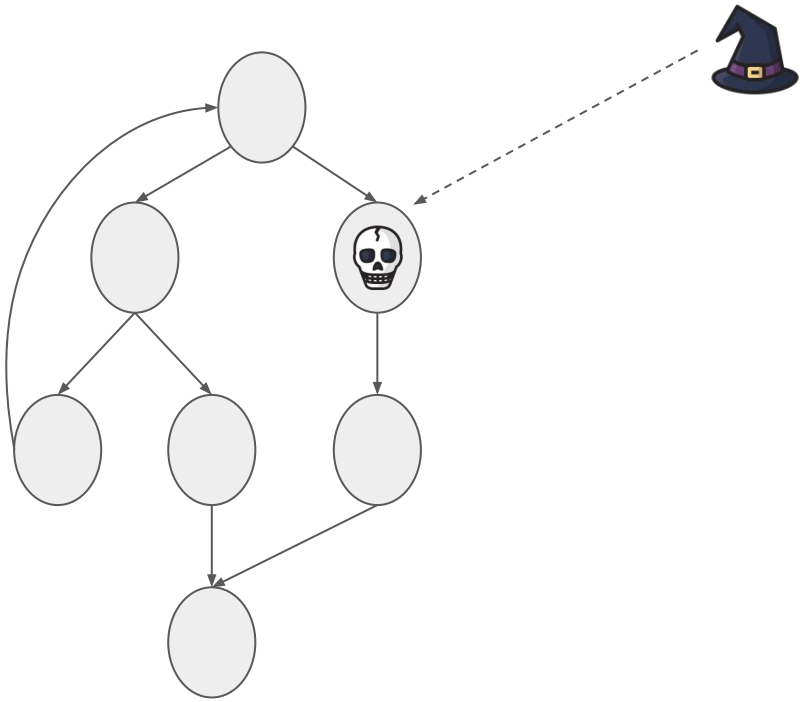


Control Flow Graph

 Intended flow

 Memory error

Code injection attacks



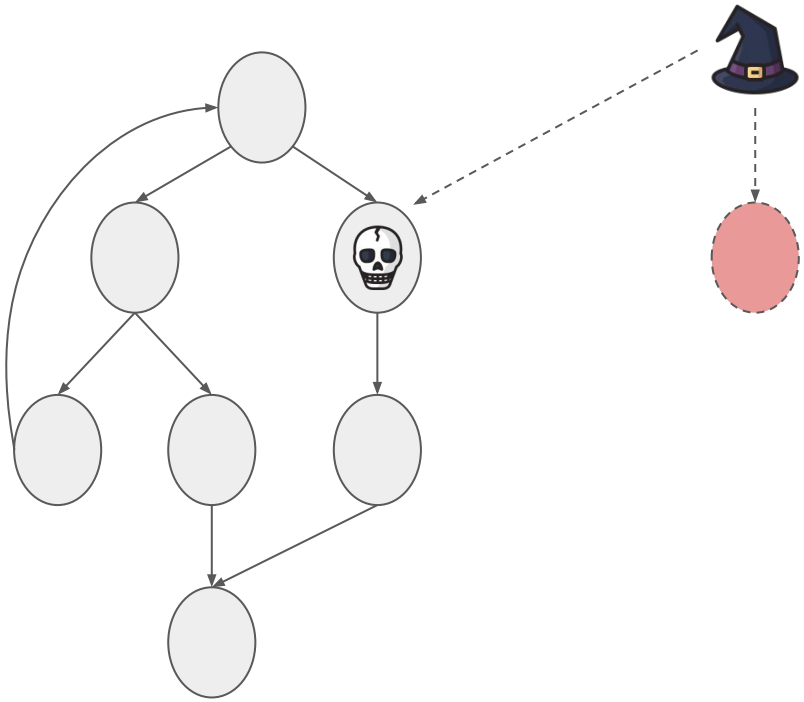
Control Flow Graph

Intended flow

Memory error

Attacker

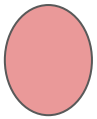
Code injection attacks



Control Flow Graph

Intended flow

Memory error

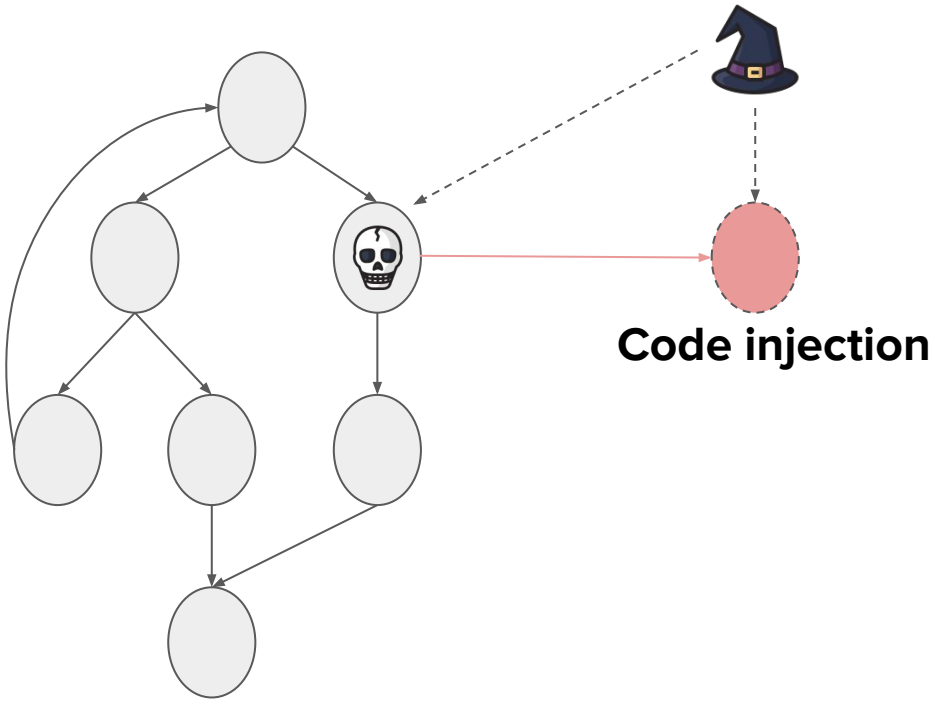


Injected code

Actual flow

Attacker

Code injection attacks



Control Flow Graph

Intended flow

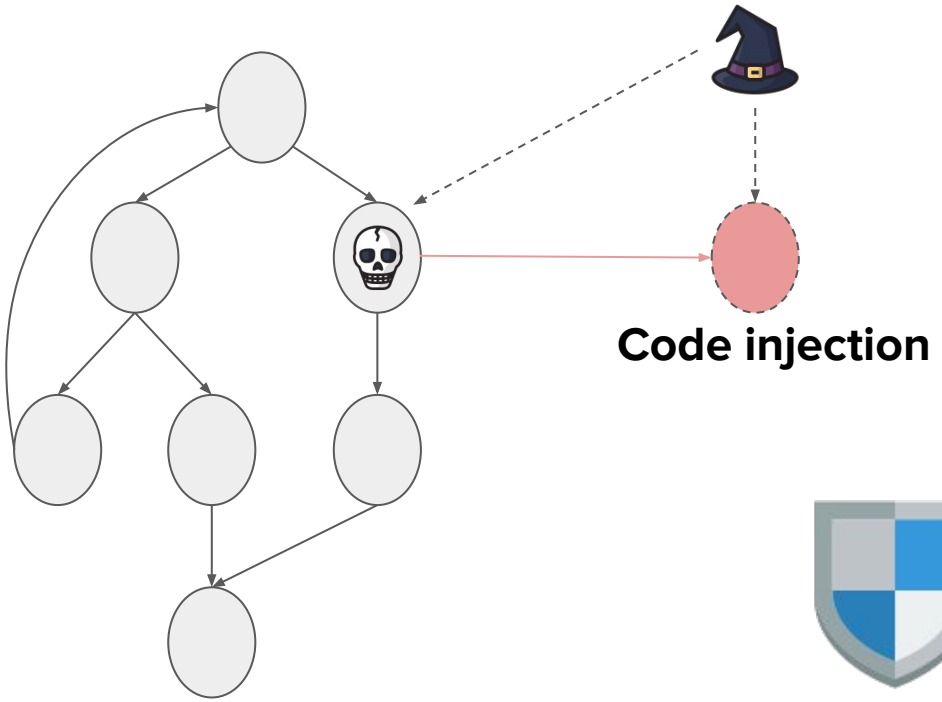
Actual flow

Memory error

Attacker

Injected code

Code injection attacks



Control Flow Graph

Data Execution Prevention (DEP) / Write XOR Execute (W⊕E)

Canaries

Intended flow

Actual flow

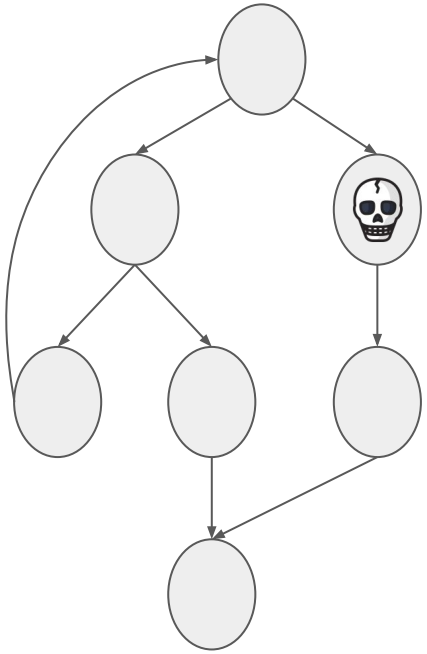
Memory error

Attacker

Injected code

Code reuse attacks

Code reuse attacks

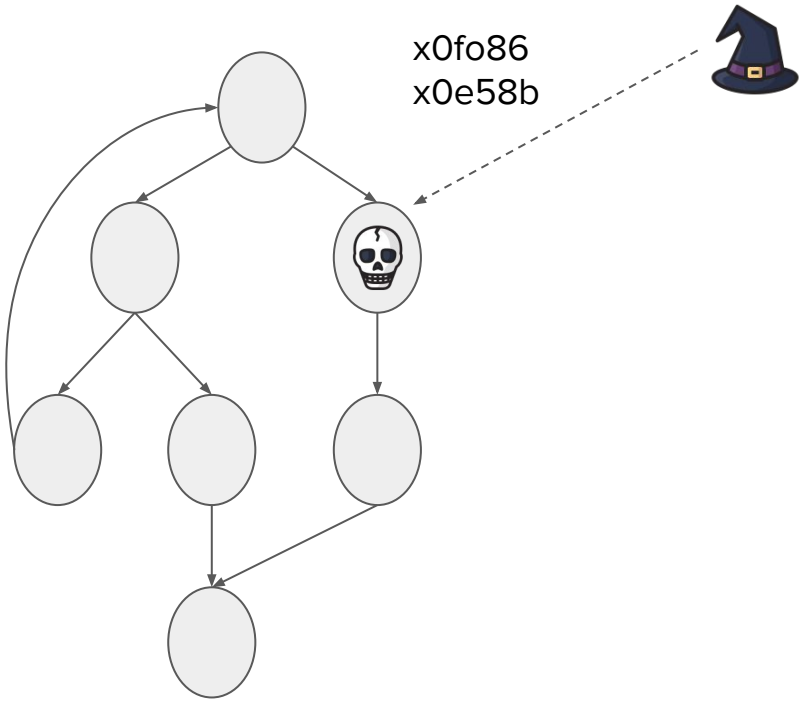


Control Flow Graph

 Intended flow

 Memory error

Code reuse attacks



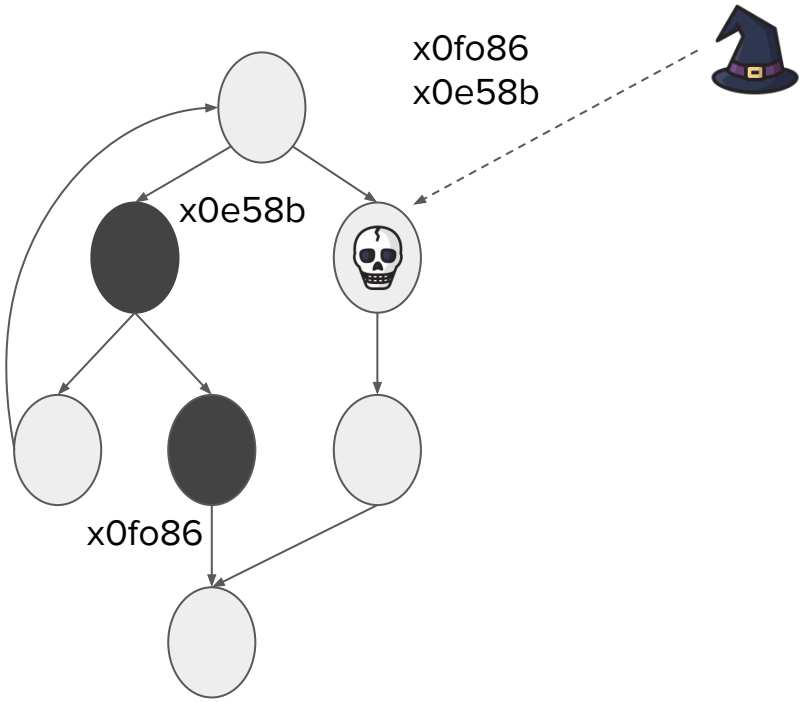
Control Flow Graph

Intended flow

Memory error

Attacker

Code reuse attacks



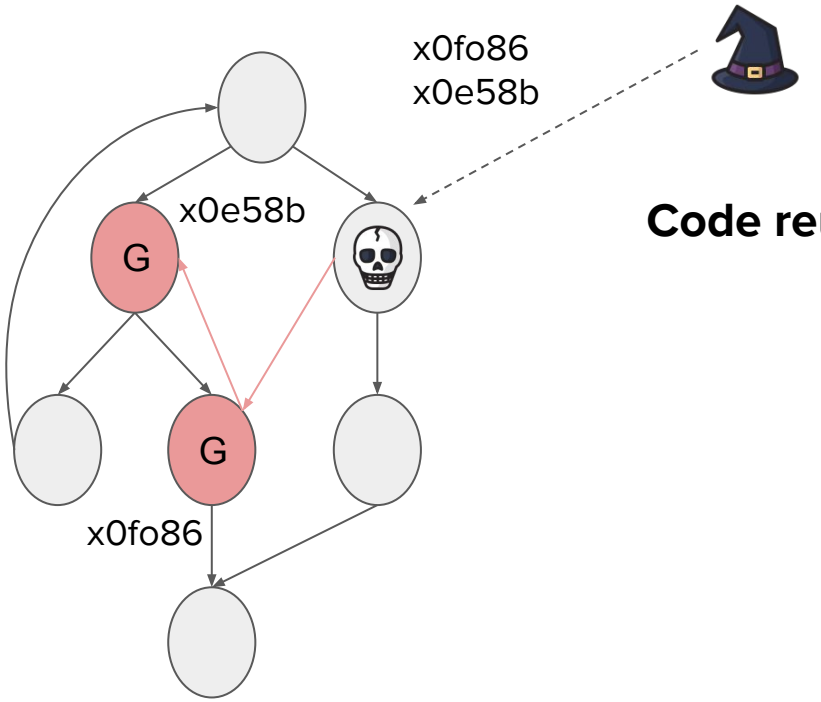
Control Flow Graph

Intended flow

Memory error

Attacker

Code reuse attacks



Code reuse

Control Flow Graph

Intended flow

Actual flow

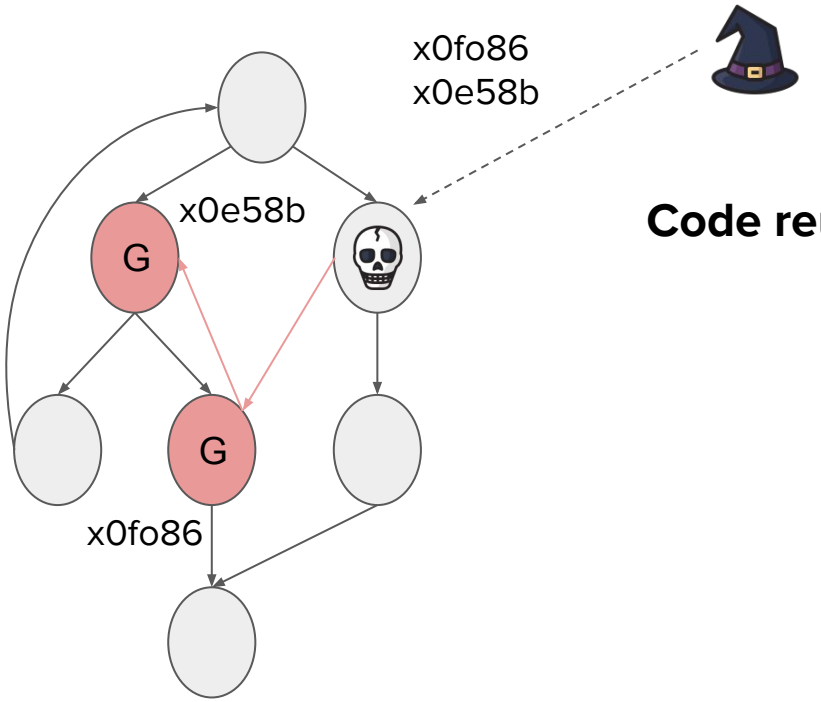
Memory error

Attacker

G

Gadget

Code reuse attacks



Control Flow Graph

Code reuse

ASLR
Kernel ASLR
➜ **Statistical**

Intended flow

Actual flow

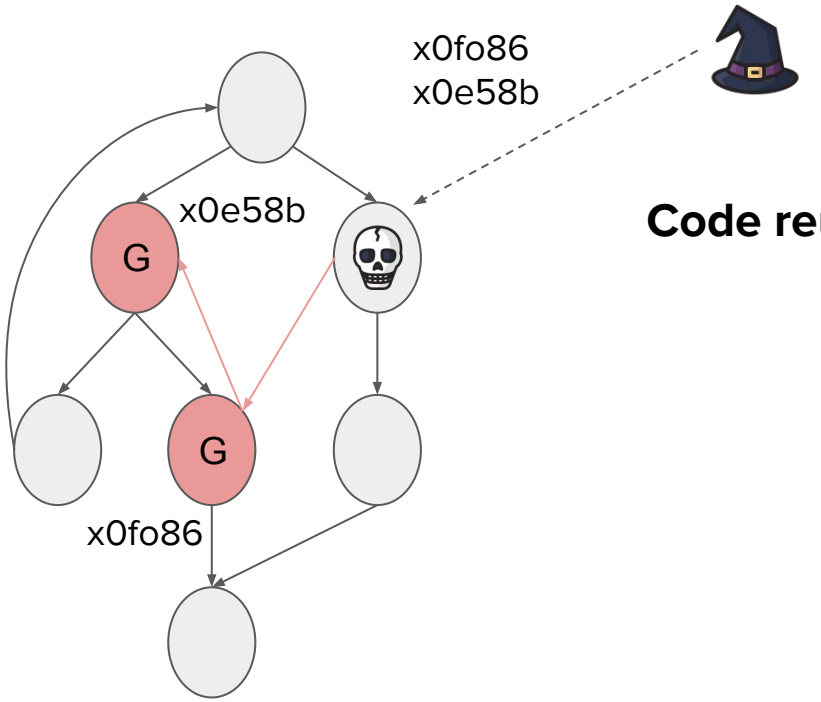
Memory error

Attacker

G

Gadget

Code reuse attacks



Control Flow Graph

Code reuse

ASLR
Kernel ASLR
➜ **Statistical**

Control Flow Integrity (CFI)

Intended flow

Actual flow

Memory error

Attacker

G

Gadget

Control-Flow Integrity (CFI)

Abadi et al. CCS'05

Control-Flow Integrity (CFI)

Abadi et al. CCS'05

1 - Offline: CFG computation

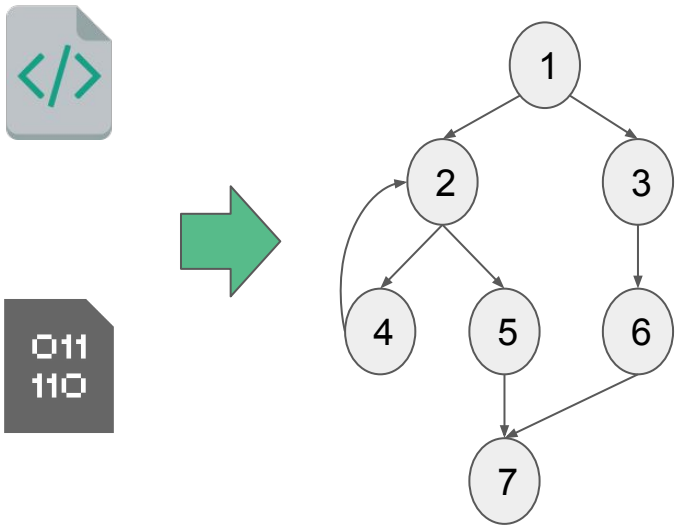
Control-Flow Integrity (CFI)

Abadi et al. CCS'05

1 - Offline: CFG computation

Control-Flow Integrity (CFI)

Abadi et al. CCS'05

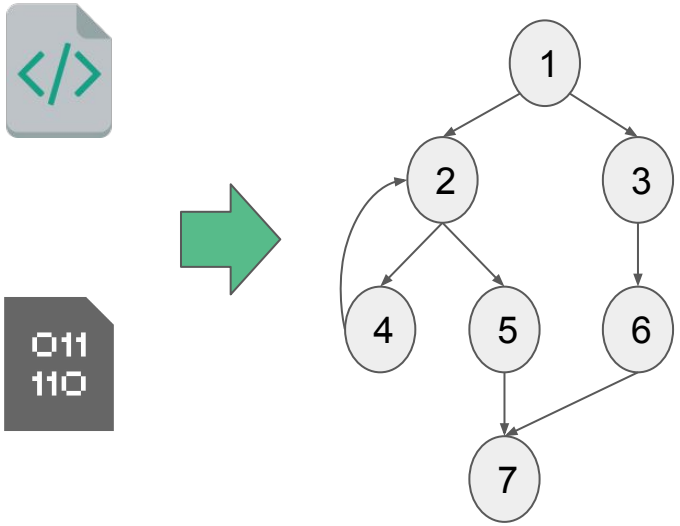


Control Flow Graph (CFG)

1 - Offline: CFG computation

Control-Flow Integrity (CFI)

Abadi et al. CCS'05



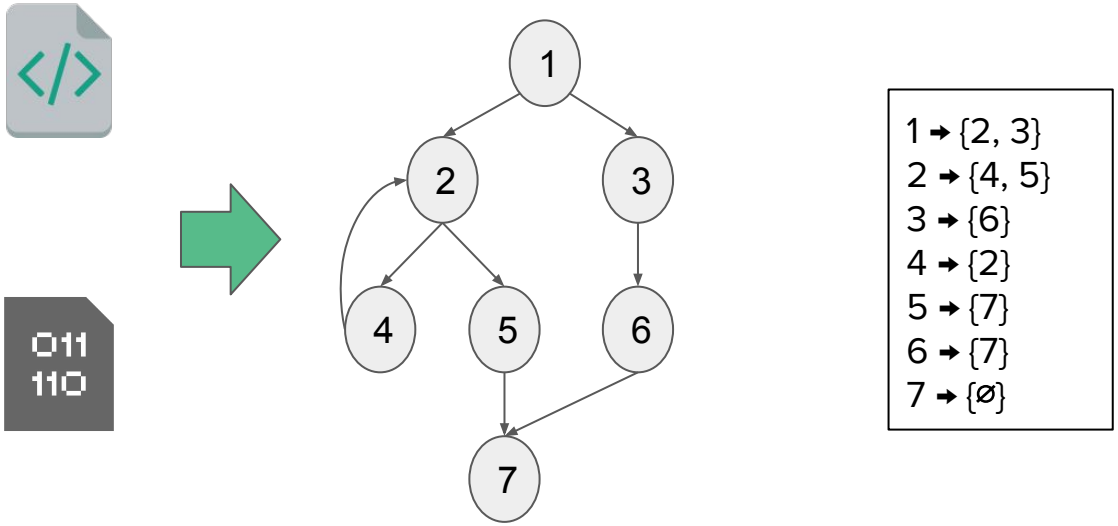
Control Flow Graph (CFG)

1 - Offline: CFG computation

2 - Runtime: CFG enforcement

Control-Flow Integrity (CFI)

Abadi et al. CCS'05



Control Flow Graph (CFG)

1 → {2, 3}
2 → {4, 5}
3 → {6}
4 → {2}
5 → {7}
6 → {7}
7 → {ø}

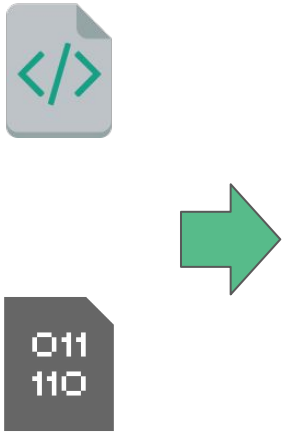
Enforced CFG

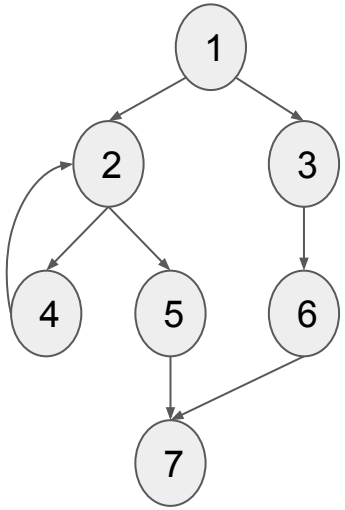
1 - Offline: CFG computation

2 - Runtime: CFG enforcement

Control-Flow Integrity (CFI)

Abadi et al. CCS'05

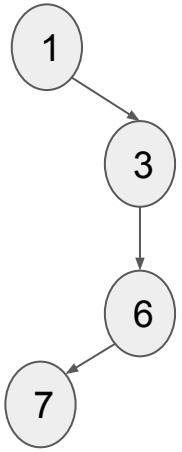




Control Flow Graph (CFG)

1 → {2, 3}
2 → {4, 5}
3 → {6}
4 → {2}
5 → {7}
6 → {7}
7 → {ø}

Enforced CFG



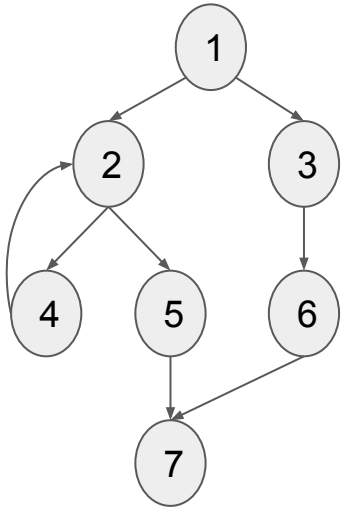
Execution 1

1 - Offline: CFG computation

2 - Runtime: CFG enforcement

Control-Flow Integrity (CFI)

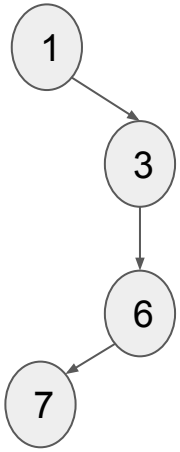
Abadi et al. CCS'05



Control Flow Graph (CFG)

1 → {2, 3}
2 → {4, 5}
3 → {6}
4 → {2}
5 → {7}
6 → {7}
7 → {ø}

Enforced CFG



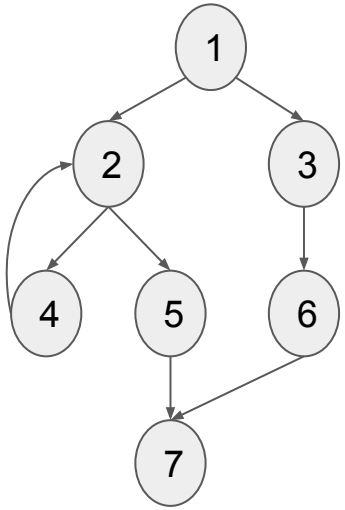
Execution 1

1 - Offline: CFG computation

2 - Runtime: CFG enforcement

Control-Flow Integrity (CFI)

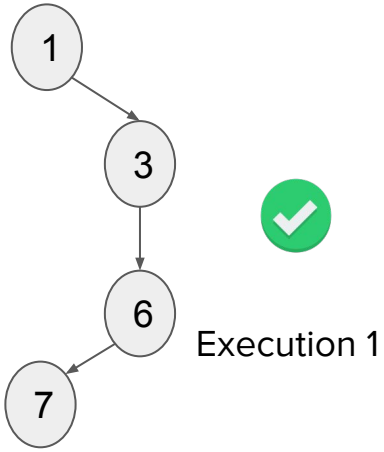
Abadi et al. CCS'05



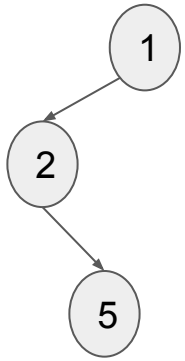
Control Flow Graph (CFG)

1 → {2, 3}
2 → {4, 5}
3 → {6}
4 → {2}
5 → {7}
6 → {7}
7 → {ø}

Enforced CFG



Execution 1



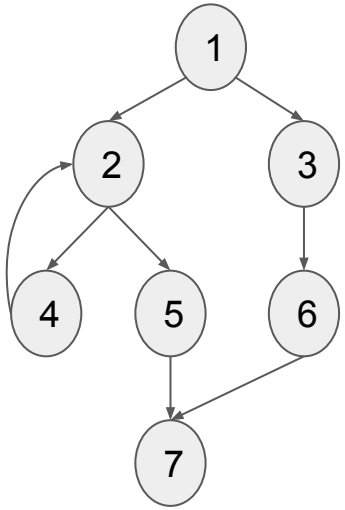
Execution 2

1 - Offline: CFG computation

2 - Runtime: CFG enforcement

Control-Flow Integrity (CFI)

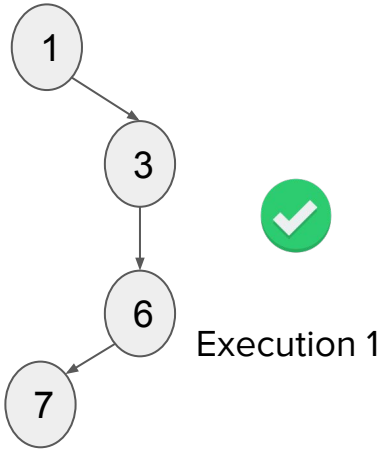
Abadi et al. CCS'05



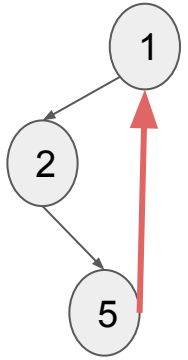
Control Flow Graph (CFG)

1 → {2, 3}
2 → {4, 5}
3 → {6}
4 → {2}
5 → {7}
6 → {7}
7 → {ø}

Enforced CFG



Execution 1



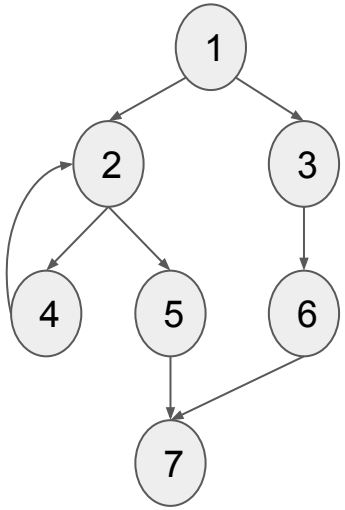
Execution 2

1 - Offline: CFG computation

2 - Runtime: CFG enforcement

Control-Flow Integrity (CFI)

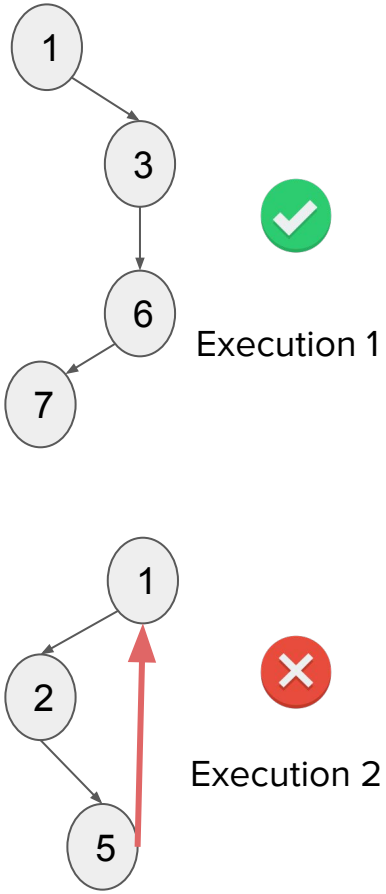
Abadi et al. CCS'05



Control Flow Graph (CFG)

1 → {2, 3}
2 → {4, 5}
3 → {6}
4 → {2}
5 → {7}
6 → {7}
7 → {ø}

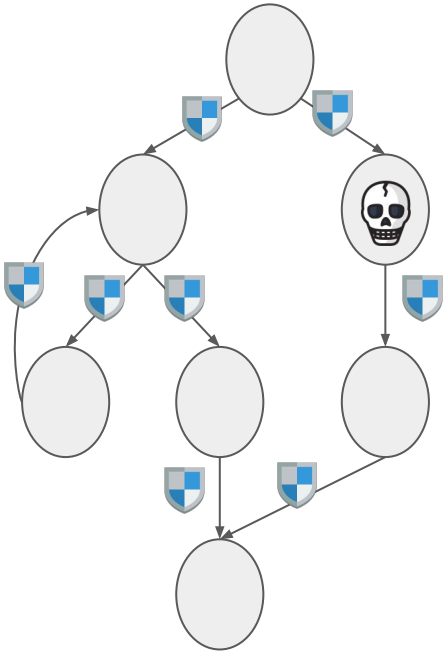
Enforced CFG



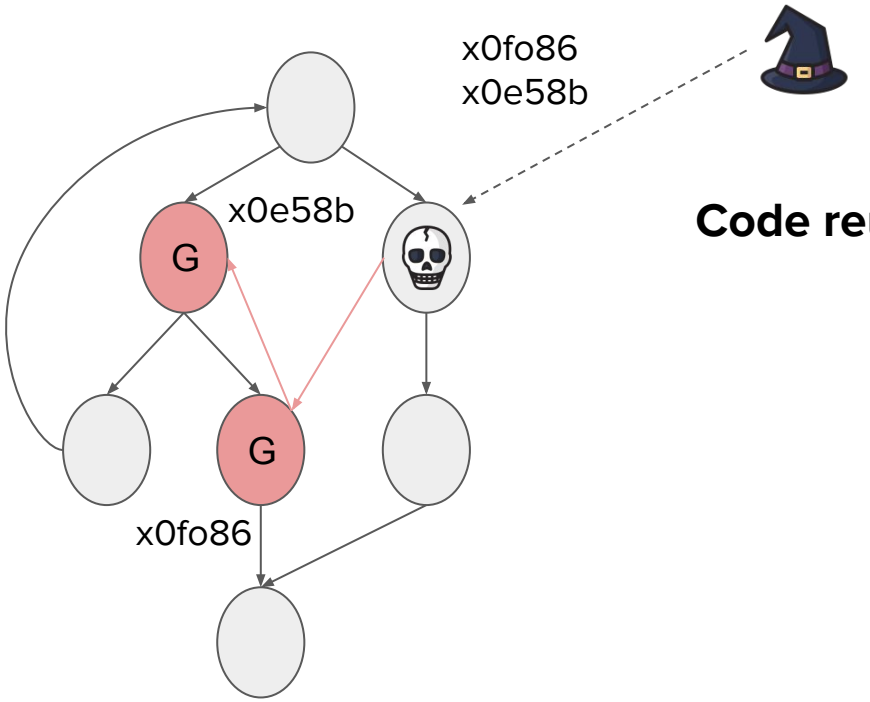
1 - Offline: CFG computation

2 - Runtime: CFG enforcement

Control-Flow Integrity (CFI) - II



Original CFG



Code reuse

Attacker's goal execution

Intended flow

Actual flow

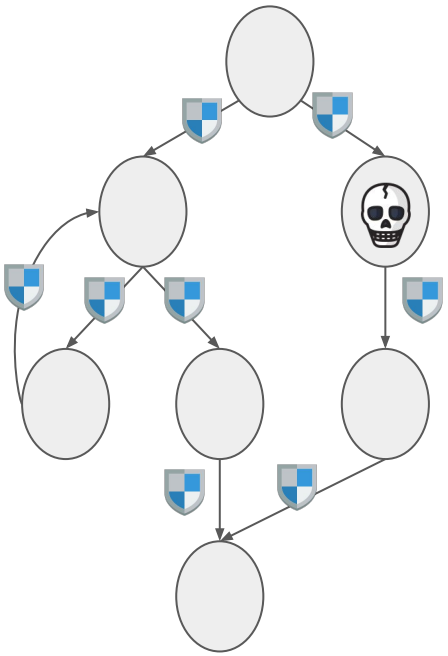
Memory error

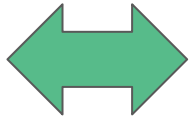
Attacker

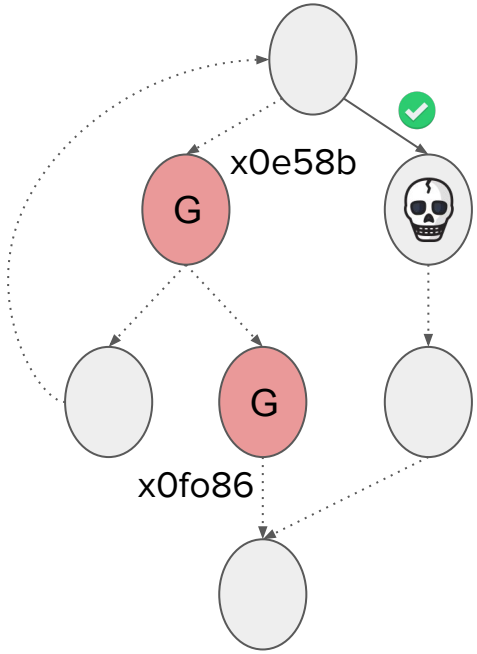
G

Gadget

Control-Flow Integrity (CFI) - II







Original CFG

Intended flow

Actual flow

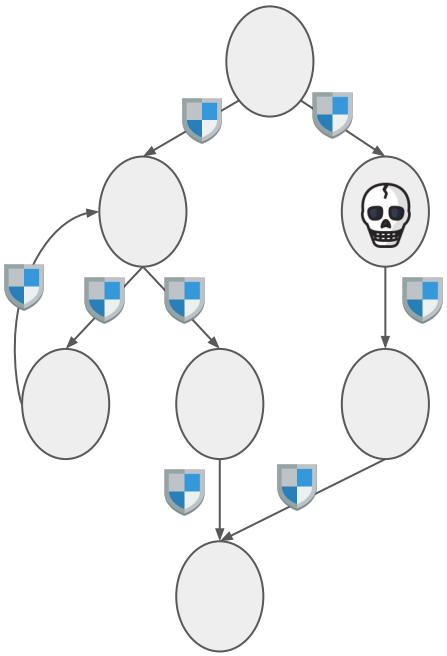
Memory error

Attacker

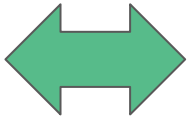
G

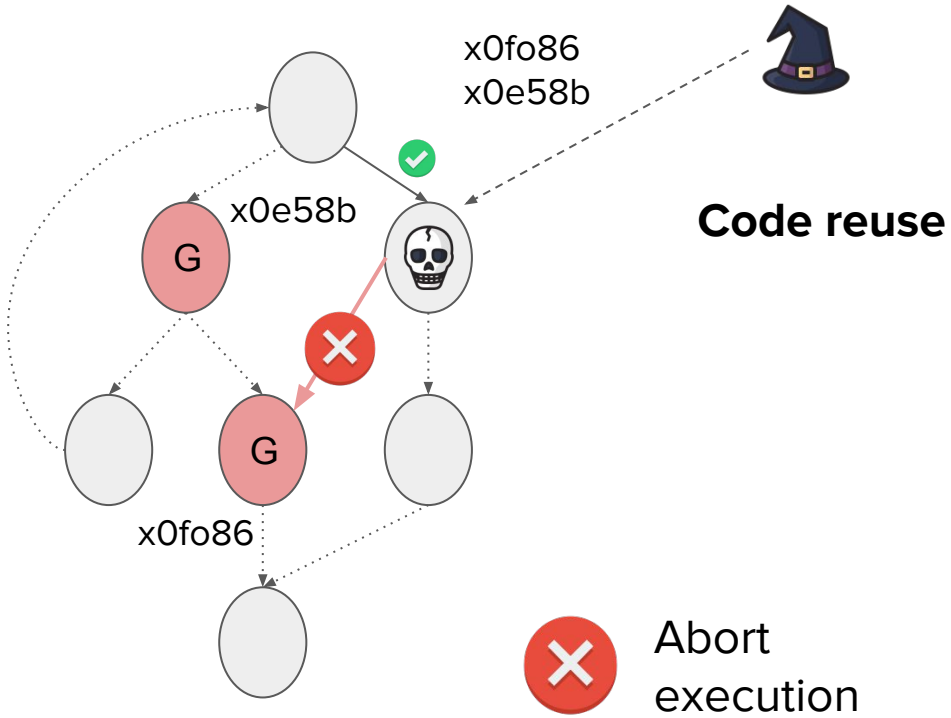
Gadget

Control-Flow Integrity (CFI) - II



Original CFG





Intended flow

Actual flow

Memory error

Attacker

G Gadget

CFI Internals

CFI Internals

Computation phase

Flow sensitive VS flow insensitive

CFI Internals

Computation phase

Flow sensitive VS flow insensitive

```
1 | obj = &x;
2 |
3 | obj = &y;
```

CFI Internals

Computation phase

Flow sensitive VS flow insensitive

```
1 | obj = &x;
2 |
3 | obj = &y;
```

obj at 1 → {x}

obj at 2→ {y}

Sensitive

obj → {x, y}

Insensitive

CFI Internals

Computation phase

Flow sensitive VS flow insensitive

```
1 | obj = &x;
2 |
3 | obj = &y;
```

obj at 1 → {x}

obj at 2→ {y}

Sensitive

obj → {x, y}

Insensitive

Context sensitive VS context insensitive

CFI Internals

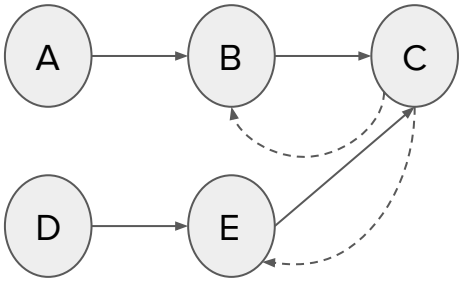
Computation phase

Flow sensitive VS flow insensitive

```
1 | obj = &x;
2 |
3 | obj = &y;
```

obj at 1 ➔ {x} obj at 2➔ {y} **Sensitive**	obj ➔ {x, y} **Insensitive**

Context sensitive VS context insensitive



Where can we return to from function C?

CFI Internals

Computation phase

Flow sensitive VS flow insensitive

```
1 | obj = &x;
2 |
3 | obj = &y;
```

obj at 1 ➔ {x}

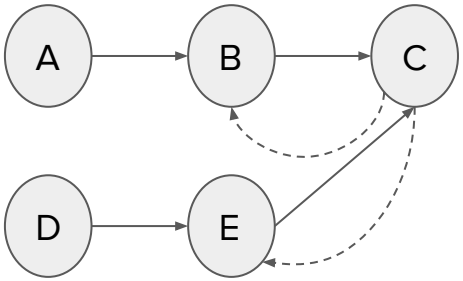
obj at 2➔ {y}

Sensitive

obj ➔ {x, y}

Insensitive

Context sensitive VS context insensitive



Where can we return to from function C?

➔ B & E

Insensitive

➔ B if called from B,
E if called from E:

Sensitive

CFI Internals

Computation phase

Flow sensitive VS flow insensitive

```
1 | obj = &x;
2 |
3 | obj = &y;
```

obj at 1 ➔ {x}

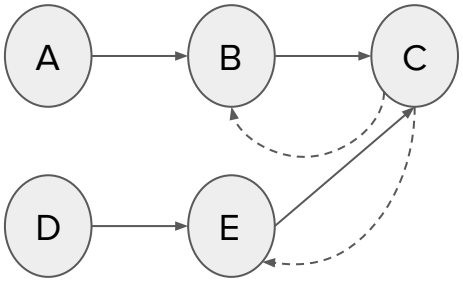
obj at 2➔ {y}

Sensitive

obj ➔ {x, y}

Insensitive

Context sensitive VS context insensitive



Where can we return to from function C?

➔ B & E

Insensitive

➔ B if called from B,
E if called from E:

Sensitive

Enforcement phase

Forward vs backward control-flow transfers

CFI Internals

Computation phase

Flow sensitive VS flow insensitive

```
1 | obj = &x;
2 |
3 | obj = &y;
```

obj at 1 ➔ {x}

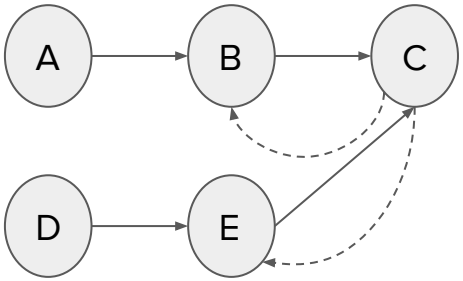
obj at 2➔ {y}

Sensitive

obj ➔ {x, y}

Insensitive

Context sensitive VS context insensitive



➔ B & E

Insensitive

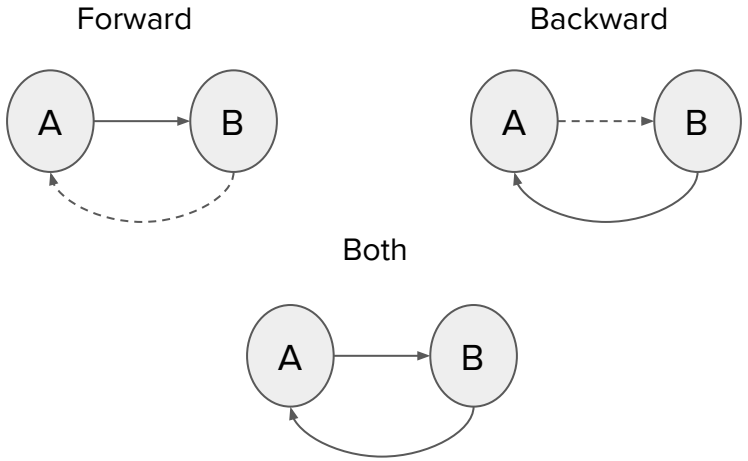
➔ B if called from B,
E if called from E:

Sensitive

Where can we return to from function C?

Enforcement phase

Forward vs backward control-flow transfers



CFI Internals

Computation phase

Flow sensitive VS flow insensitive

```
1 | obj = &x;
2 |
3 | obj = &y;
```

obj at 1 ➔ {x}

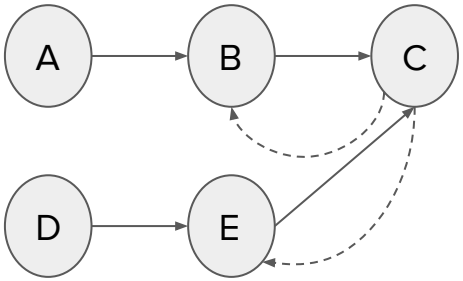
obj at 2 ➔ {y}

Sensitive

obj ➔ {x, y}

Insensitive

Context sensitive VS context insensitive



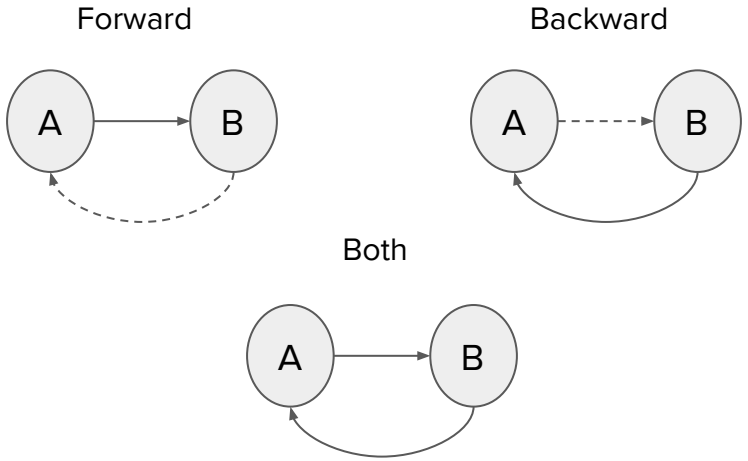
➔ B & E
Insensitive

➔ B if called from B,
E if called from E:
Sensitive

Where can we return to from function C?

Enforcement phase

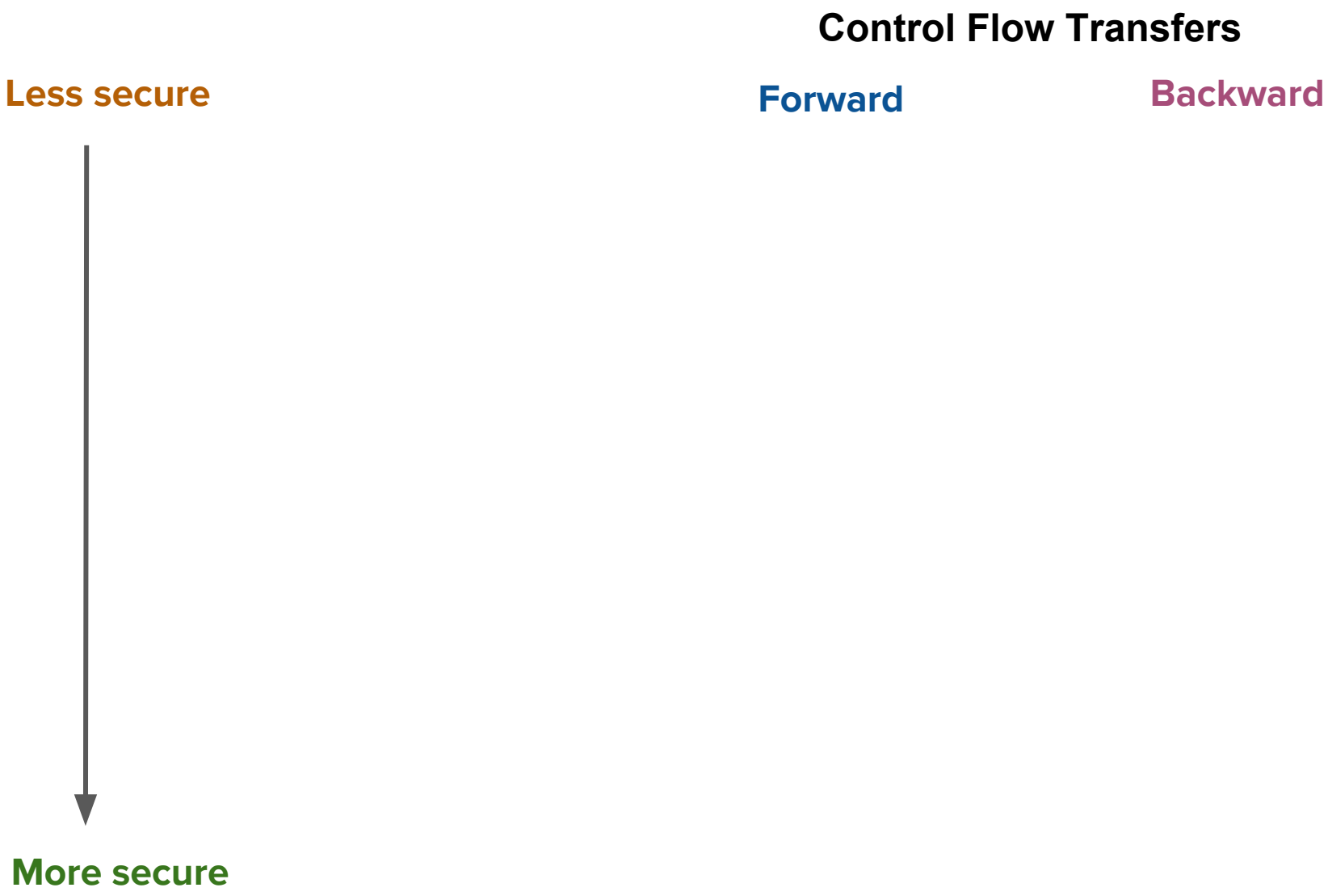
Forward vs backward control-flow transfers



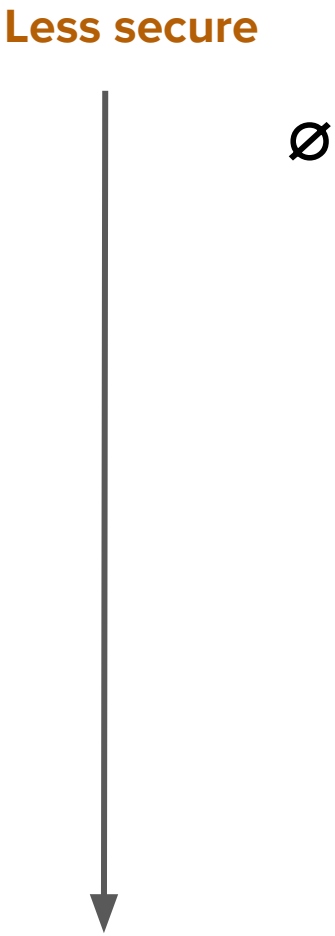
Which control-flow transfers do we take into account?

Comparison fields

Comparison fields



Comparison fields



∅

Control Flow Transfers

Forward **Backward**

Every control flow transfer is allowed

Comparison fields

Less secure

More secure

∅

Equivalent classes

Control Flow Transfers

Forward **Backward**

Every control flow transfer is allowed

Makes the assumption that two destinations are equivalent if they come from the same source

Comparison fields

Less secure

More secure

∅

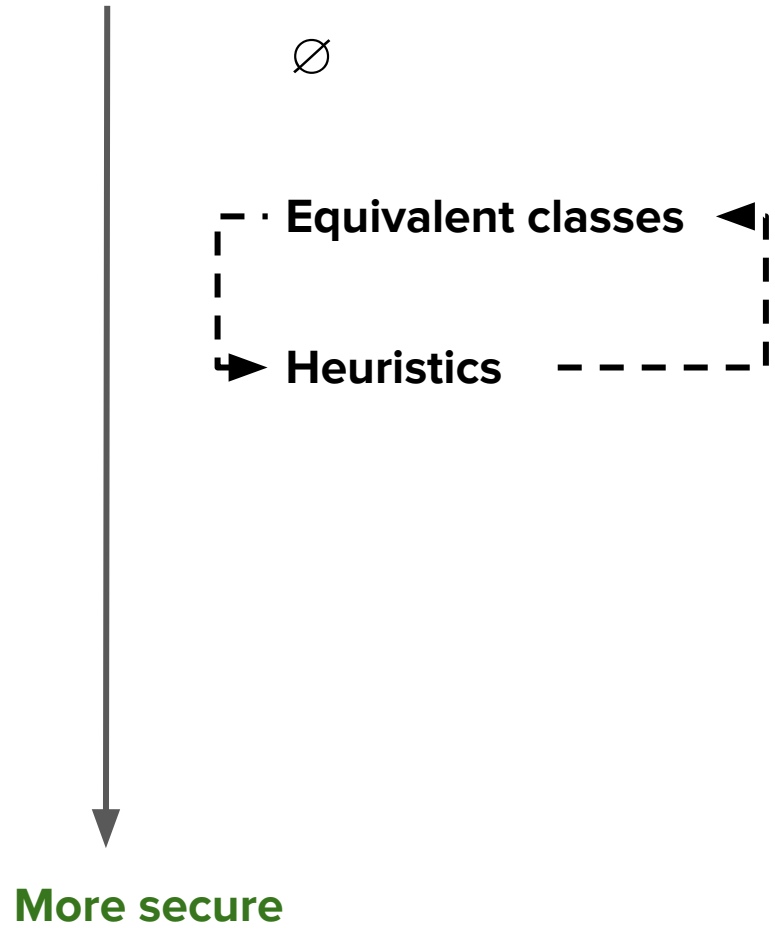
Equivalent classes

Heuristics

Control Flow Transfers

Forward	**Backward**
Every control flow transfer is allowed	
Makes the assumption that two destinations are equivalent if they come from the same source	
May or may not work on specially crafted cases	

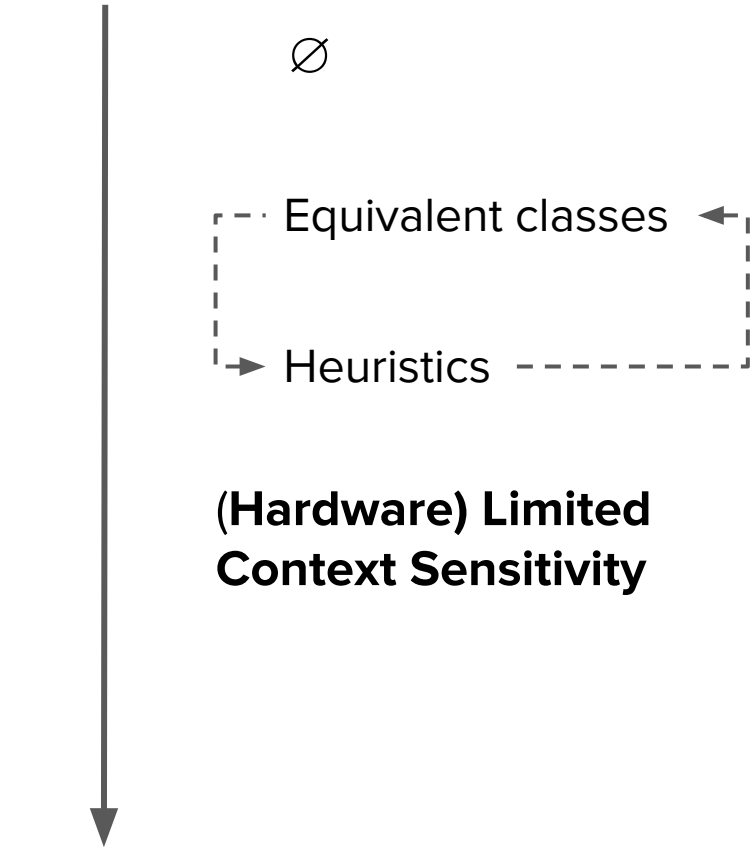
Comparison fields



Control Flow Transfers

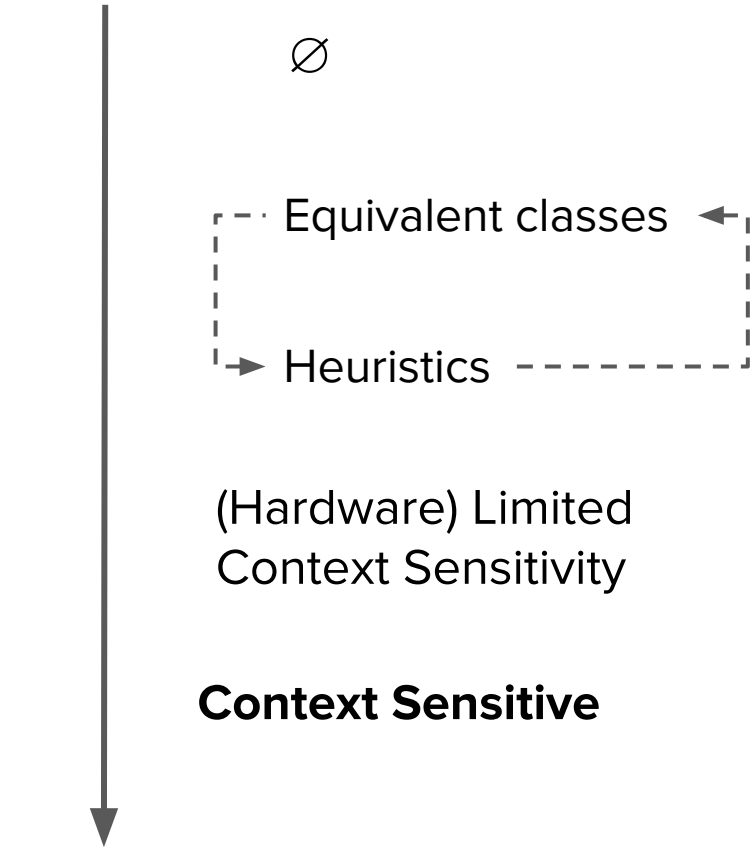
Forward	Backward
Every control flow transfer is allowed	
Makes the assumption that two destinations are equivalent if they come from the same source	
May or may not work on specially crafted cases	

Comparison fields



Control Flow Transfers	
Forward	**Backward**
Every control flow transfer is allowed	
Makes the assumption that two destinations are equivalent if they come from the same source	
May or may not work on specially crafted cases	
There is some restriction on full CS	

Comparison fields



Control Flow Transfers	
Forward	**Backward**
Every control flow transfer is allowed	
Makes the assumption that two destinations are equivalent if they come from the same source	
May or may not work on specially crafted cases	
There is some restriction on full CS	
	

Userland CFI Implementations

Original CFI, Abadi et al. CCS'05

MoCFI, Davi et al. NDSS'12

CCFIR, Zhang et al. Oakland'13

Bin-CFI, Zhang et al. Usenix Sec.'13

kBouncer, Pappas et al. Usenix Sec.'13

ROPecker, Cheng et al. NDSS'14

SafeDispatch, Jang et al. NDSS'14

MCFI, Niu & Tan. PLDI'14

RockJIT, Niu & Tan. CCS'14

O-CFI, Mohan et al. NDSS'15

PathArmor, van der Veen et al. CCS'15

VTV / IFCC, Tice et al. Usenix Sec.'15

πCFI, Niu & Tan. CCS'15

TypeArmor, van der Veen et al. Oakland'16

Userland CFI - Binary

Forward \ Backward	∅	Equivalent Classes	Heuristics	Hardware limited CS	Context Sensitive
Equivalent Classes					
Heuristics					
Hardware limited CS					

CS: Context Sensitivity

Userland CFI - Binary

Forward \ Backward	∅	Equivalent Classes	Heuristics	Hardware limited CS	Context Sensitive
Equivalent Classes	TypeArmor	CCFIR Bin-CFI O-CFI			Original CFI MoCFI
Heuristics			kBouncer		
Hardware limited CS				PathArmor	

CS: Context Sensitivity

Userland CFI - Source Code

Backward / Forward	∅	Equivalent Classes	Limited CS
Equivalent Classes			
Limited CS			
Context Sensitive			

CS: Context Sensitivity

Userland CFI - Source Code

Forward \ Backward	∅	Equivalent Classes	Limited CS
Equivalent Classes		MCFI RockJIT	
Limited CS			πCFI
Context Sensitive	VTV / IFCC SafeDispatch		

CS: Context Sensitivity

Kernel space CFI Implementations

State-based CFI (**SBCFI**), Petroni & Hicks. CCS'07

Hypersafe, Wang & Jiang. Oakland'10

kGuard, Kemerlis et al. Usenix Sec.'12

KCoFI, Criswell et al. Oakland'14

Kernel space CFI

Forward \ Backward	Exists in kernel space	Equivalent Classes	Limited CS
Exists in kernel space			
Equivalent Classes			
Limited CS			

CS: Context Sensitivity

Kernel space CFI

Forward \ Backward	Exists in kernel space	Equivalent Classes	Limited CS
Exists in kernel space	kGuard		
Equivalent Classes		KCoFI	
Limited CS			Hypersafe

CS: Context Sensitivity

CFI - Other Schemes

CFI - Other Schemes

Userland

ROPecker, Cheng et al. NDSS'14

Kernel module

Heuristics for **forward** and **backward** control flow transfers

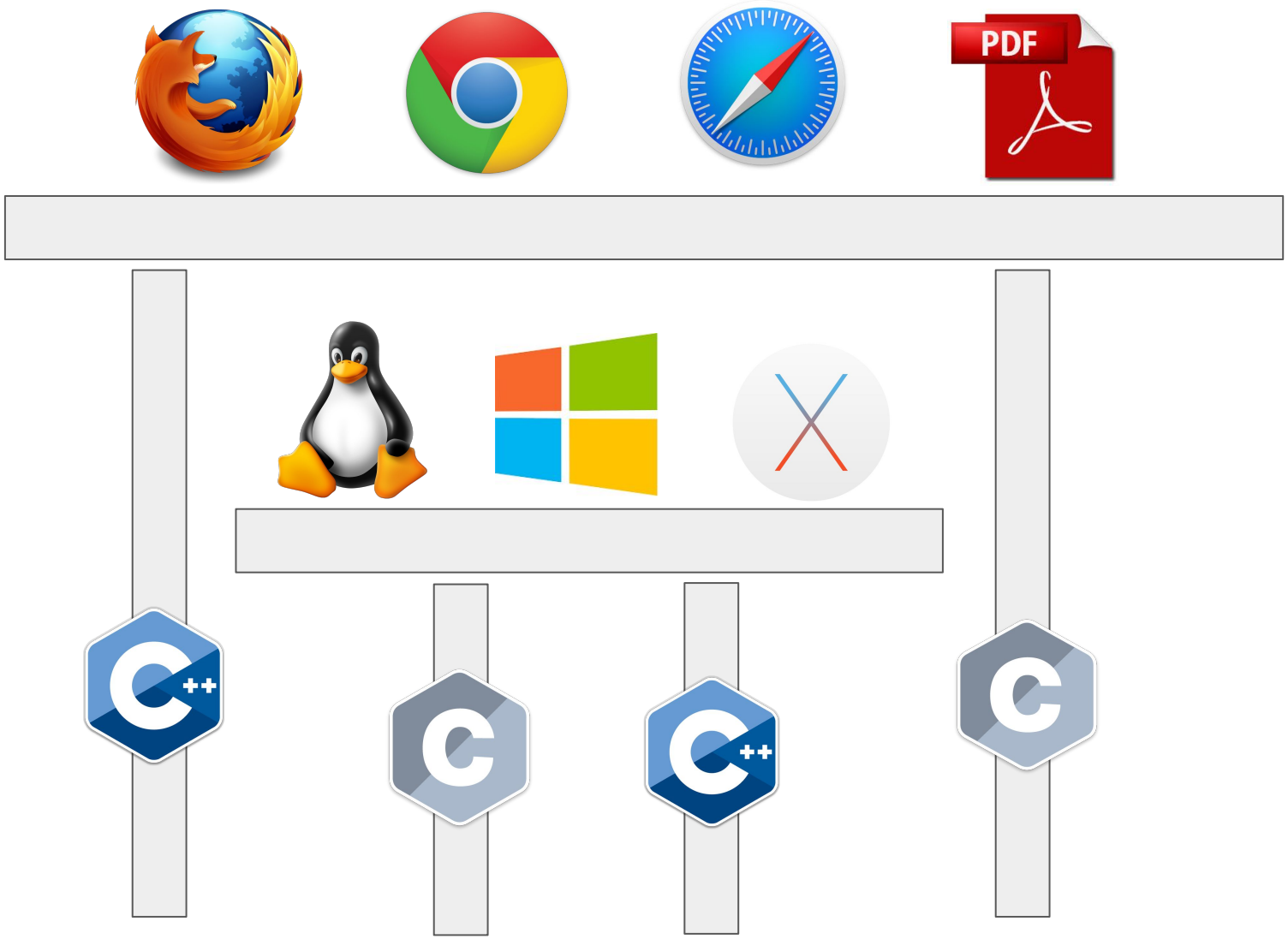
Kernel space

SBCFI, Petroni & Hicks. CCS'07

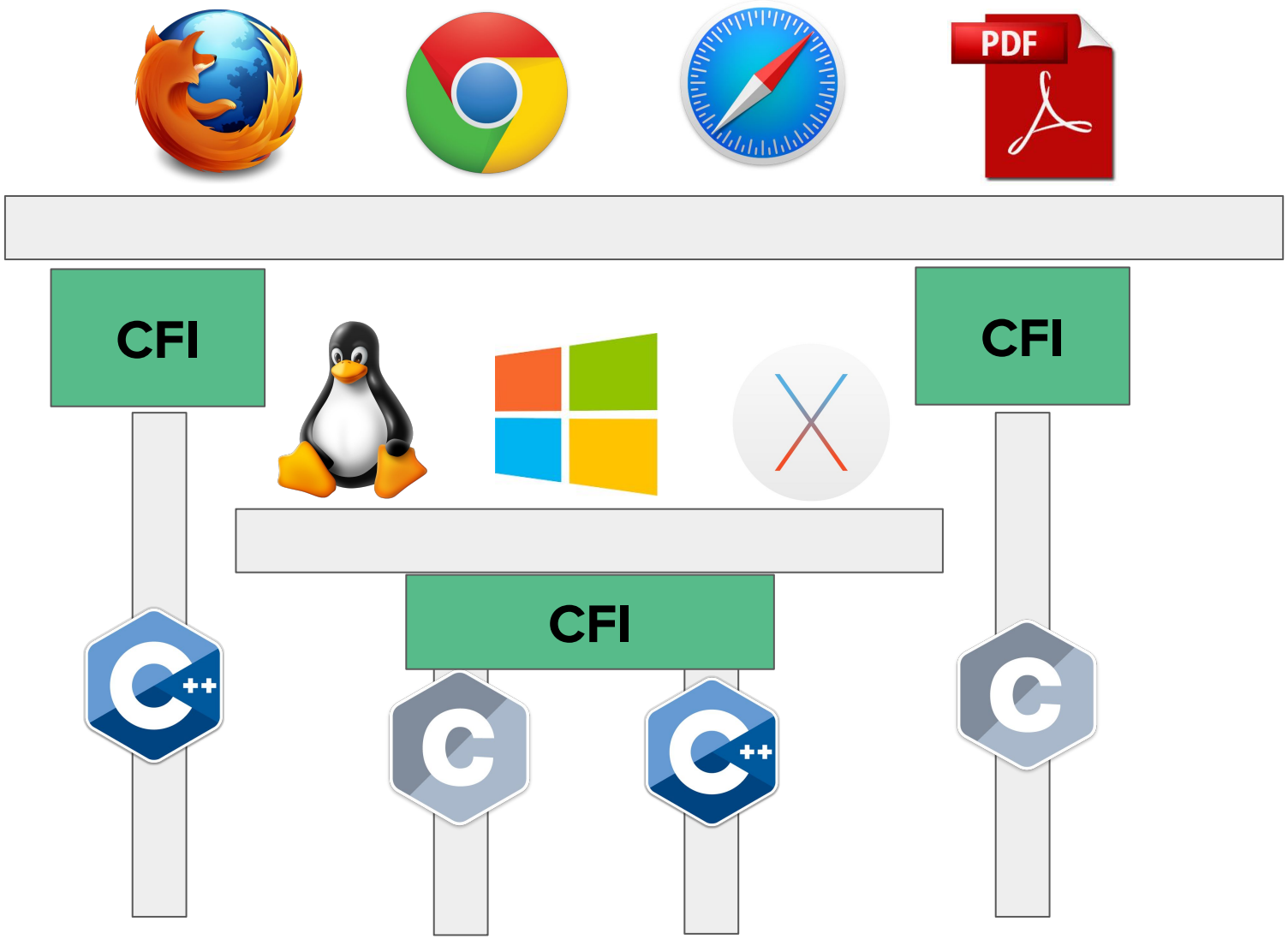
Virtual machine monitor

Forward: **compares CFGs**
Backward: ⌀

Closing remarks



Closing remarks



Feel me Flow:A Review of Control-Flow Integrity Methods for User and Kernel Space

Irene Díez-Franco, Igor Santos
DeustoTech, University of Deusto
irene.diez@deusto.es isantos@deusto.es