

Securing Resource Constrained Processors with Name Confusion

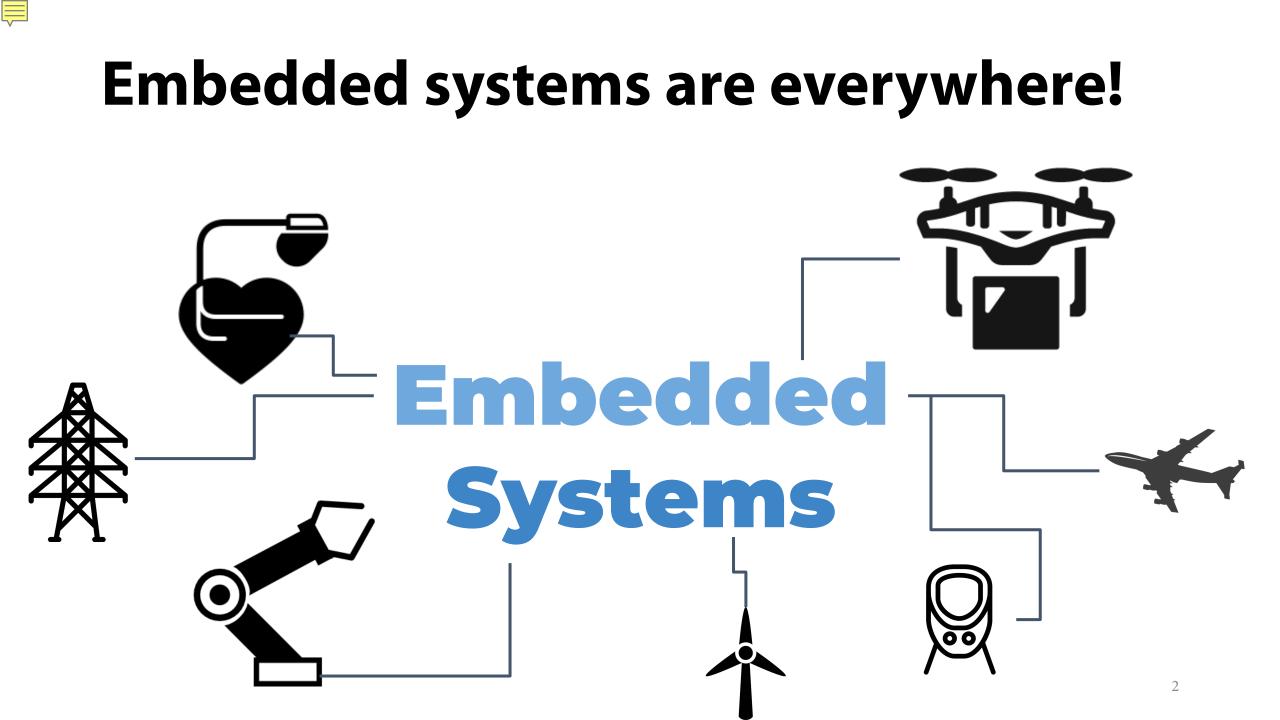
Mohamed Tarek Ibn Ziad, Miguel A. Arroyo, Evgeny Manzhosov, Vasileios P. Kemerlis, and Simha Sethumadhavan



COMPUTER SCIENCE

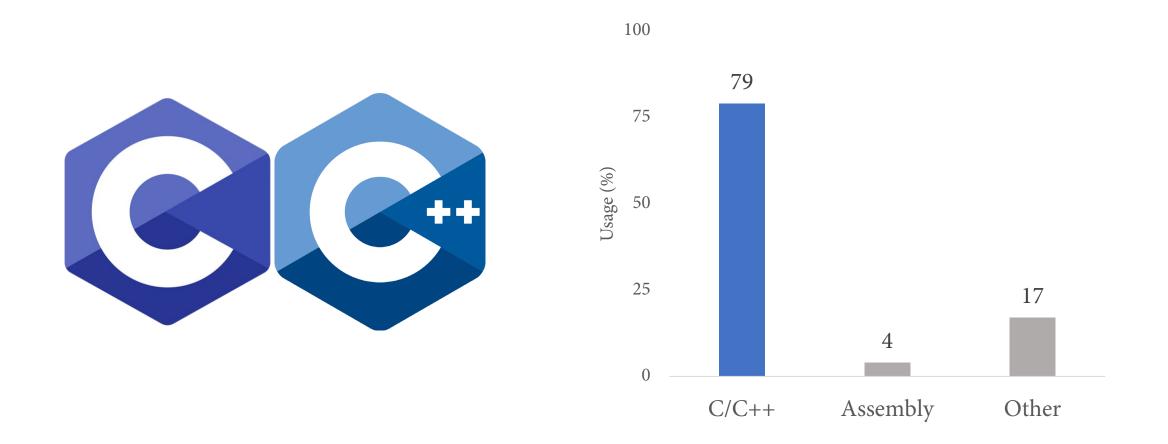


Columbia University Brown University 09/21/2021



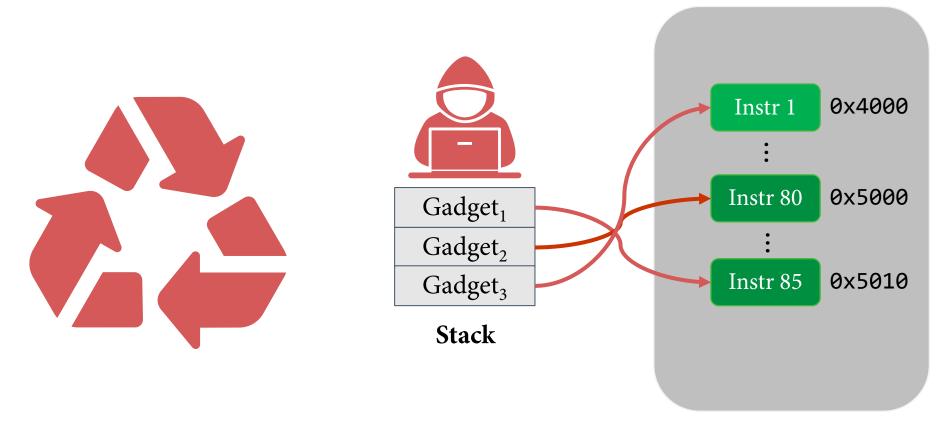
Why focus on software threats?

CPSs are predominantely written in memory unsafe languages.



Ę

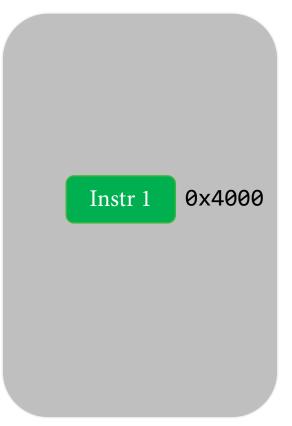
Code Reuse Attacks



Virtual Address Space

Ē

An instruction has a single name (or address).



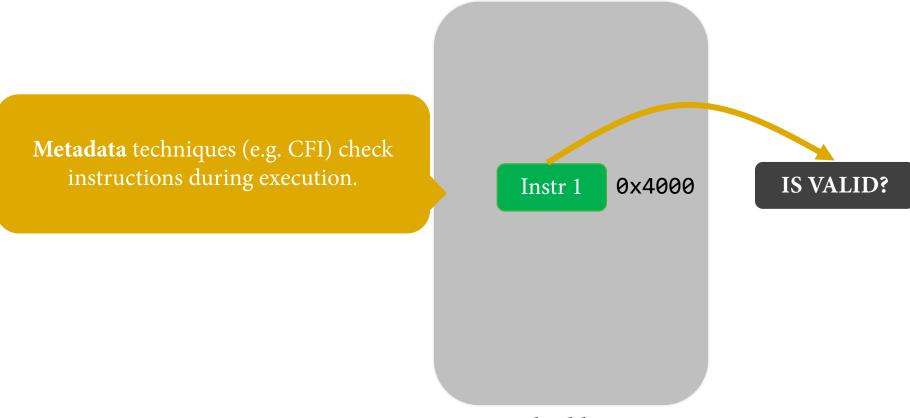
Ę

An instruction has a single name (or address).

Information hiding techniques (e.g. ISR) randomize instruction encoding. Enc(Instr 1) 0x4000

Ę

An instruction has a single name (or address).

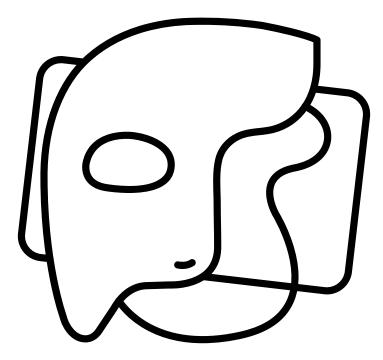


Ę

An instruction has a single name (or address).

Moving target techniques change instruction names over time. 0x4000 Instr 1 Instr 1 0x4000



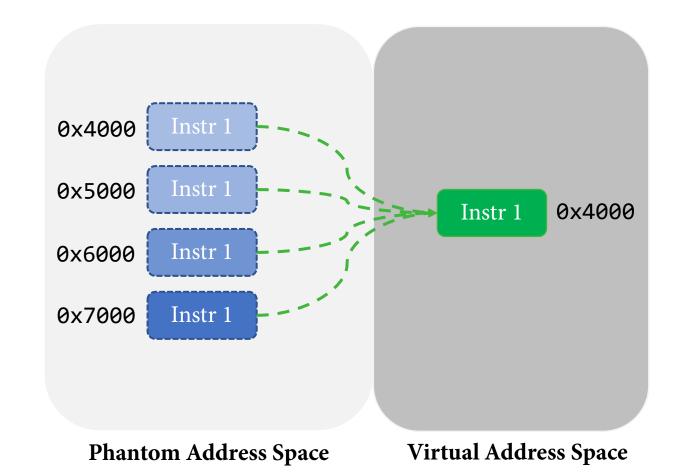


Phantom Name System A Name Confusion Architecture

Name Confusion Architecture

Ę

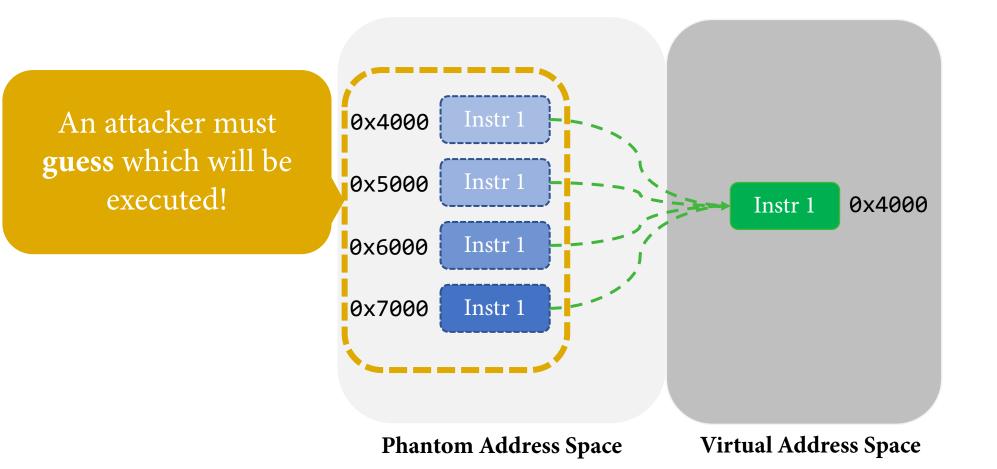
Multiple phantom addresses alias to an instruction.



Name Confusion Architecture

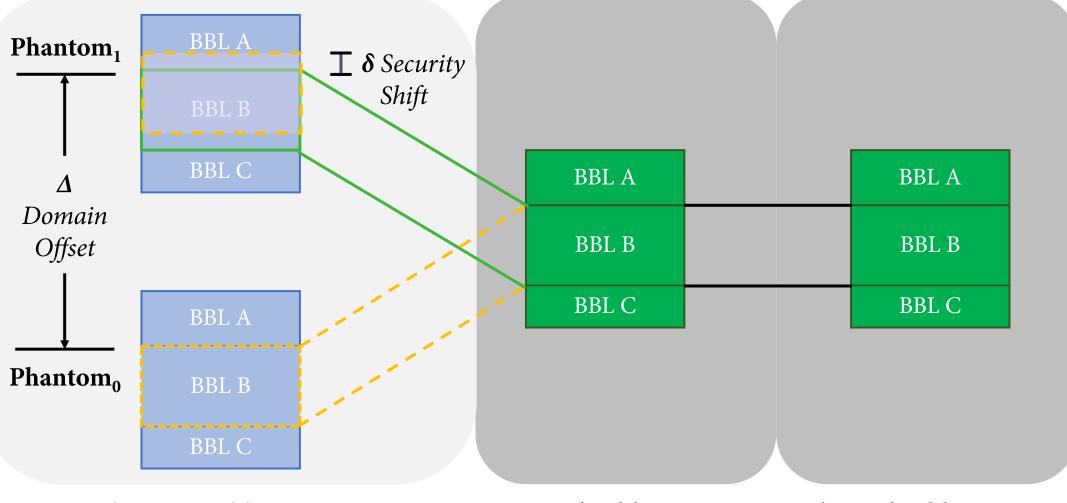
Ļ

Multiple phantom addresses alias to an instruction.



How are phantoms constructed?

Phantoms are logically displaced relative to the original program.



Phantom Address Space

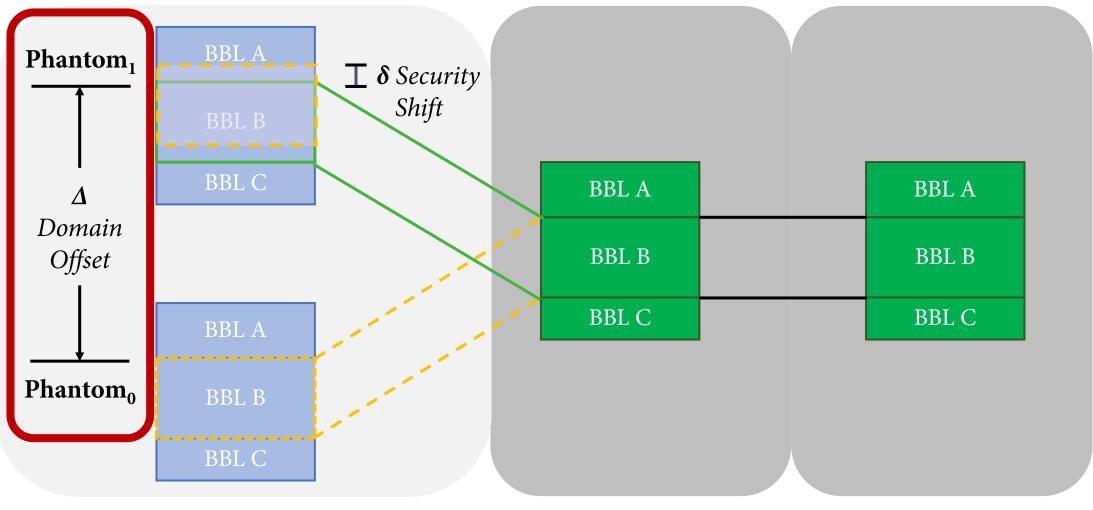
Ē

Virtual Address Space

Physical Address Space 12

How are phantoms constructed?

Phantoms are logically displaced relative to the original program.



Phantom Address Space

Ē

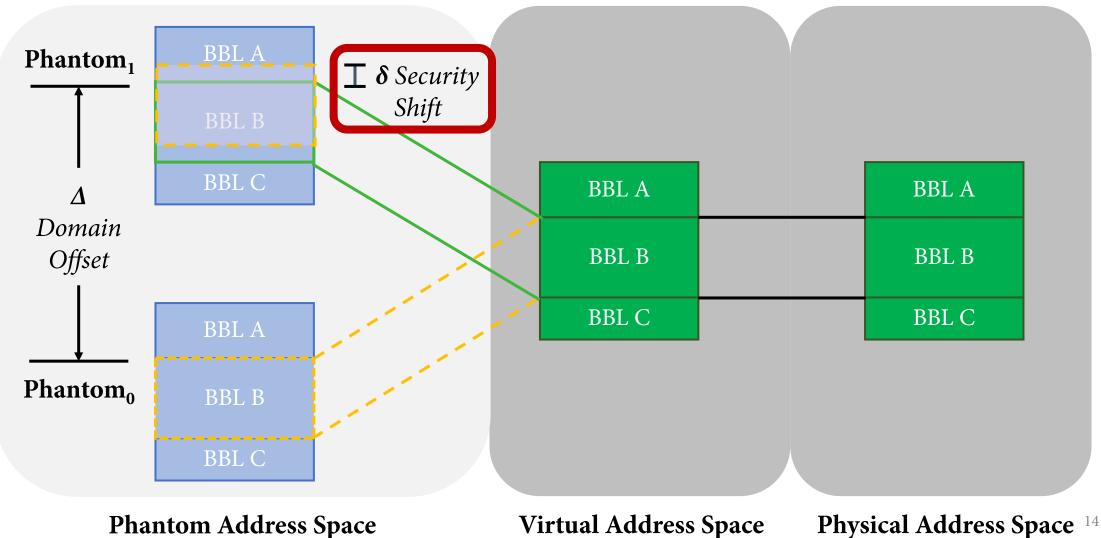
Virtual Address Space

Physical Address Space ¹³

How are phantoms constructed?

Ē

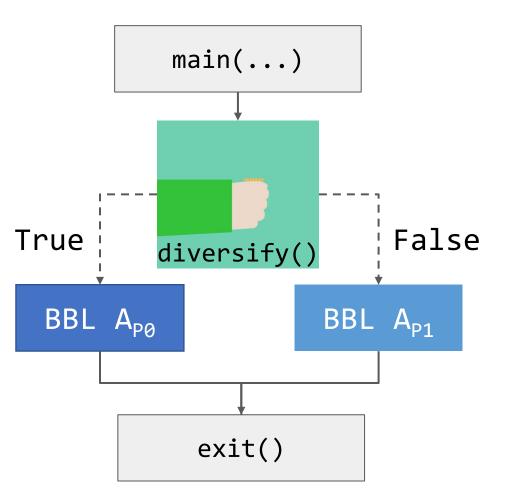
Phantoms are logically displaced relative to the original program.



How does PNS diversify execution?

It diversifies the *path* of execution at every basic block.

Ę

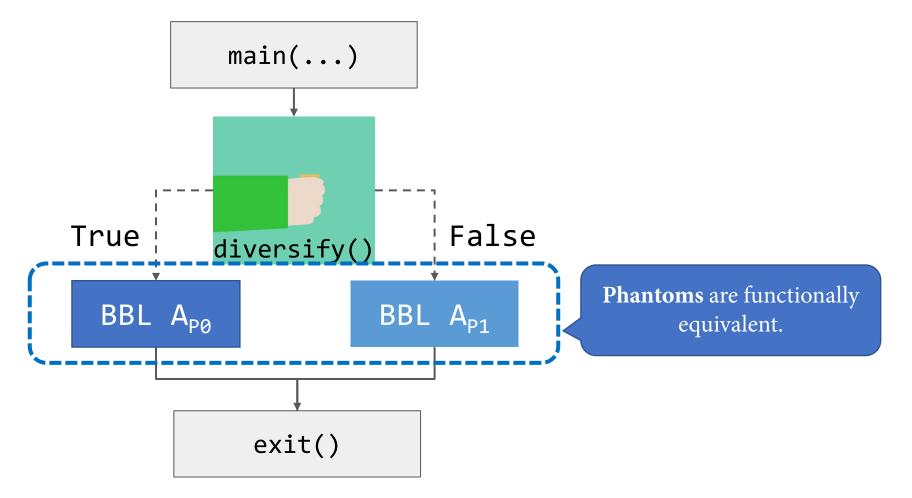


Program Control Flow Graph

How does PNS diversify execution?

It diversifies the *path* of execution at every basic block.

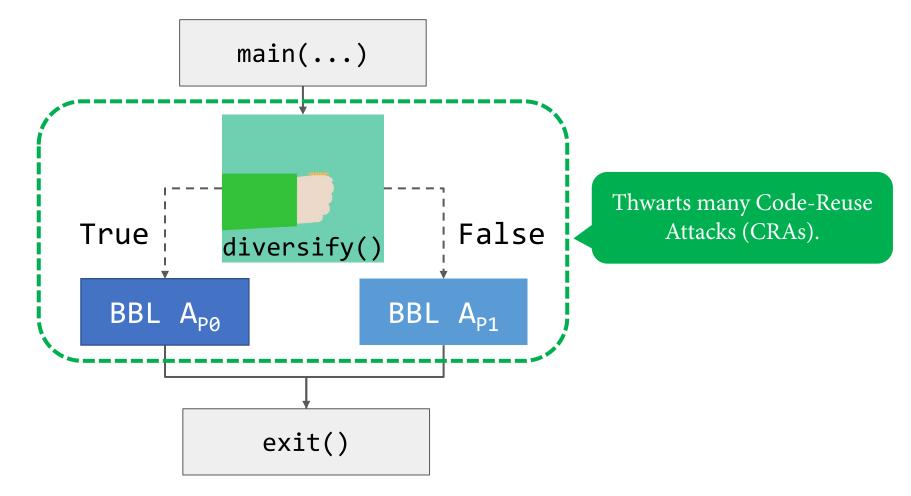
Ę



Program Control Flow Graph

How does PNS diversify execution?

It diversifies the *path* of execution at every basic block.



Program Control Flow Graph

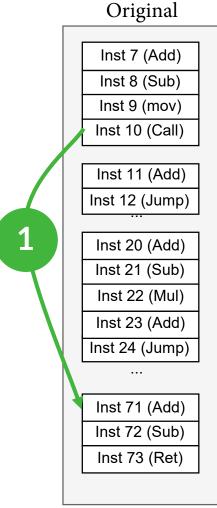
Phantoms force an adversary to guess the execution path.

Original

Ę

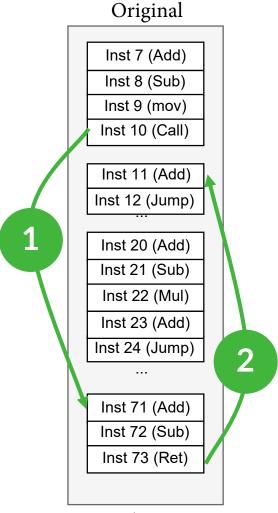
Inst 7 (Add)
Inst 8 (Sub)
Inst 9 (mov)
Inst 10 (Call)
Inst 11 (Add)
Inst 12 (Jump)
•••
Inst 20 (Add)
Inst 21 (Sub)
Inst 22 (Mul)
Inst 23 (Add)
Inst 24 (Jump)
Inst 71 (Add)
Inst 72 (Sub)

Phantoms force an adversary to guess the execution path.



Ē

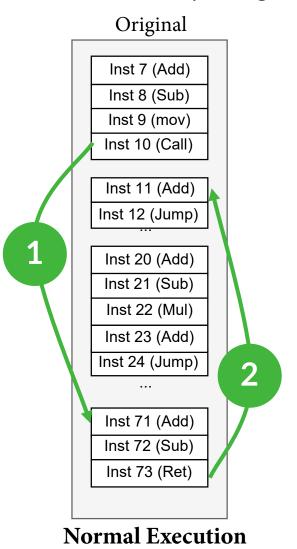
Phantoms force an adversary to guess the execution path.

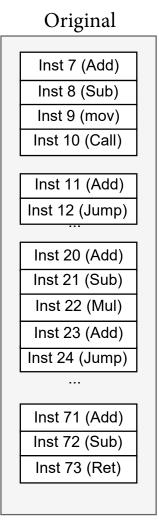


Ē

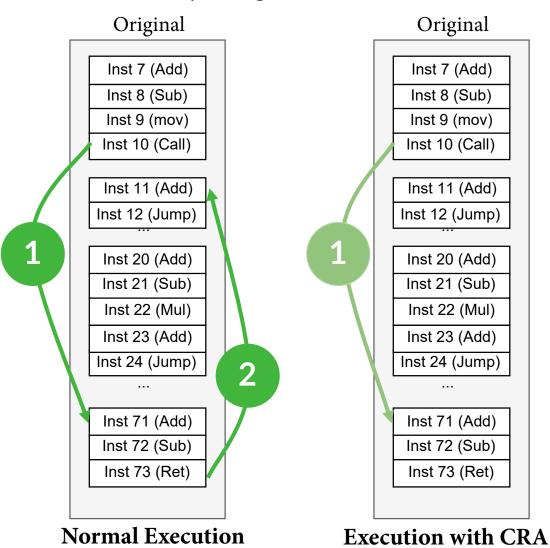
Phantoms force an adversary to guess the execution path.

Ē

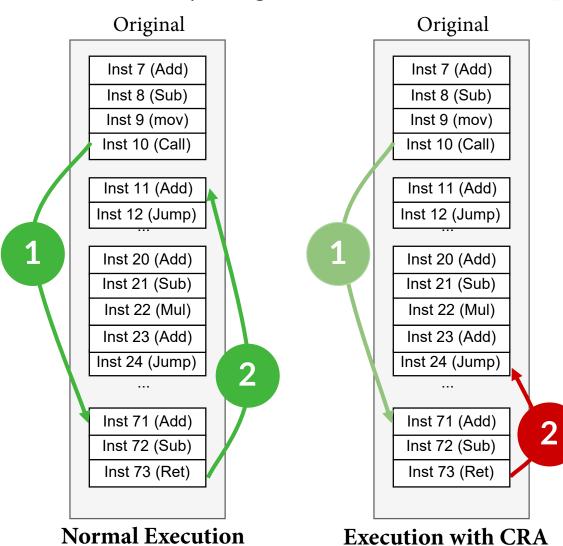




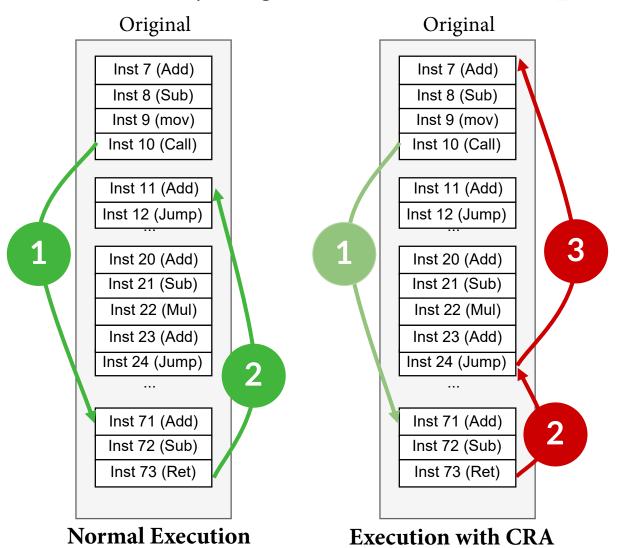
Phantoms force an adversary to guess the execution path.



Phantoms force an adversary to guess the execution path.

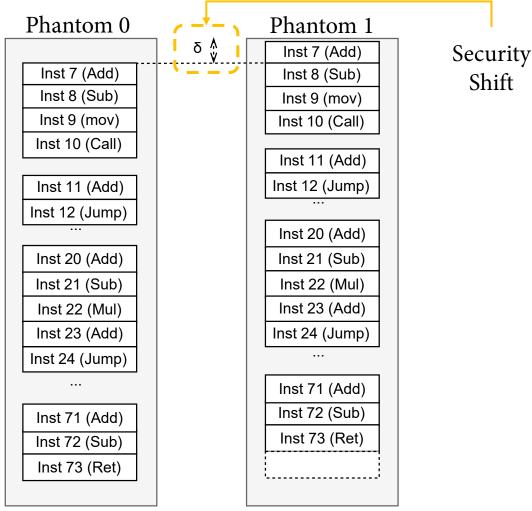


Phantoms force an adversary to guess the execution path.



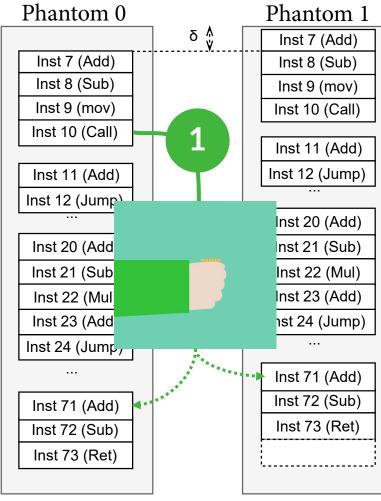
Shift

Phantoms force an adversary to guess the execution path.

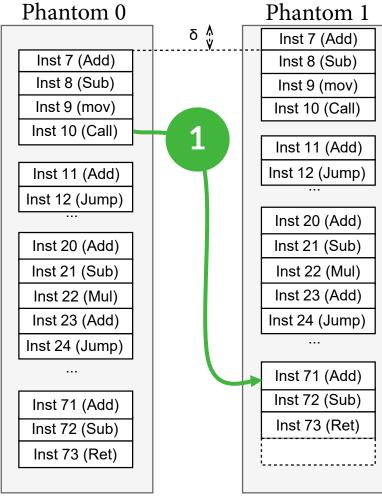


Ē

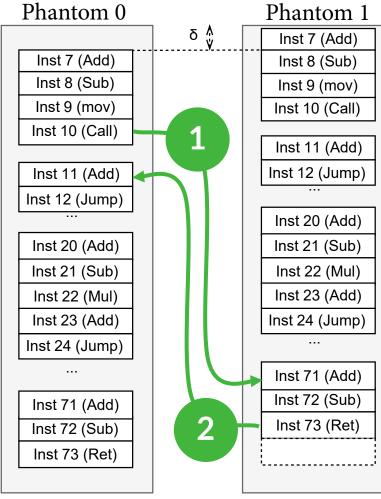
Phantoms force an adversary to guess the execution path.



Phantoms force an adversary to guess the execution path.

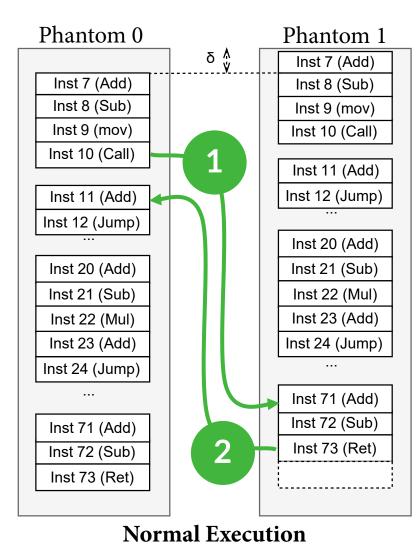


Phantoms force an adversary to guess the execution path.



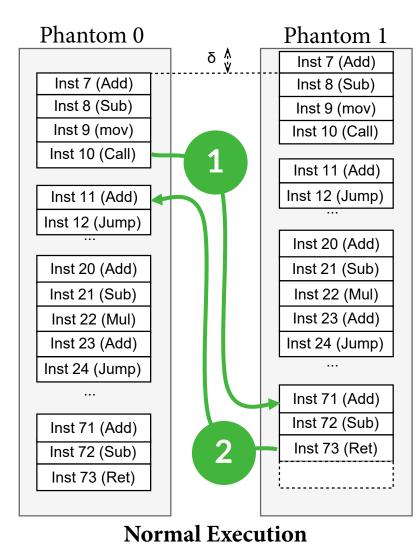
Ē

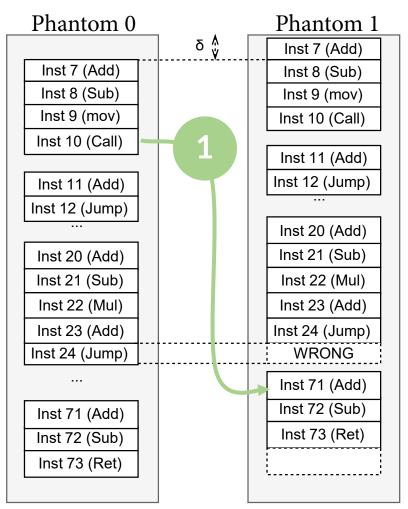
Phantoms force an adversary to guess the execution path.



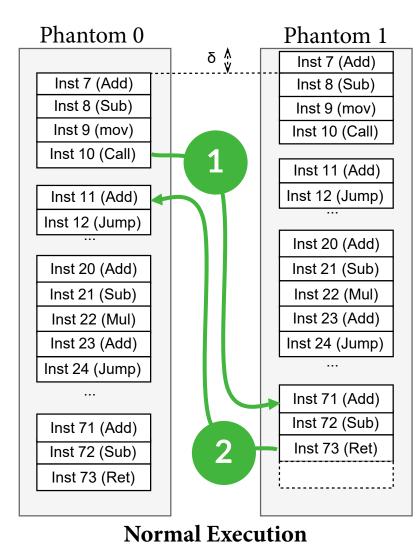
Phantom 0		Phantom 1
	δÅ	Inst 7 (Add)
Inst 7 (Add)		Inst 8 (Sub)
Inst 8 (Sub)		Inst 9 (mov)
Inst 9 (mov)		Inst 10 (Call)
Inst 10 (Call)		
		Inst 11 (Add)
Inst 11 (Add)		Inst 12 (Jump)
Inst 12 (Jump)		
		Inst 20 (Add)
Inst 20 (Add)		Inst 21 (Sub)
Inst 21 (Sub)		Inst 22 (Mul)
Inst 22 (Mul)		Inst 23 (Add)
Inst 23 (Add)		Inst 24 (Jump)
Inst 24 (Jump)		WRONG
		Inst 71 (Add)
Inst 71 (Add)		Inst 72 (Sub)
Inst 72 (Sub)		Inst 73 (Ret)
Inst 73 (Ret)		

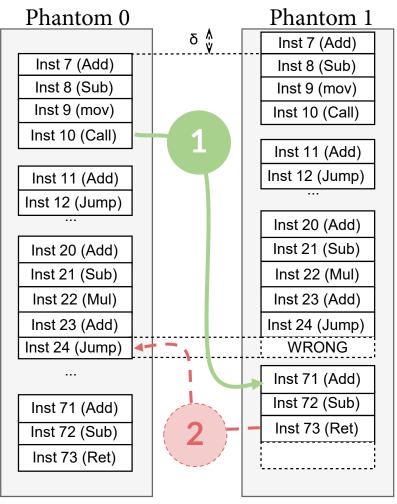
Phantoms force an adversary to guess the execution path.



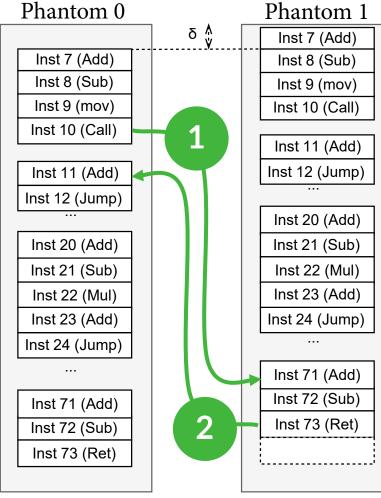


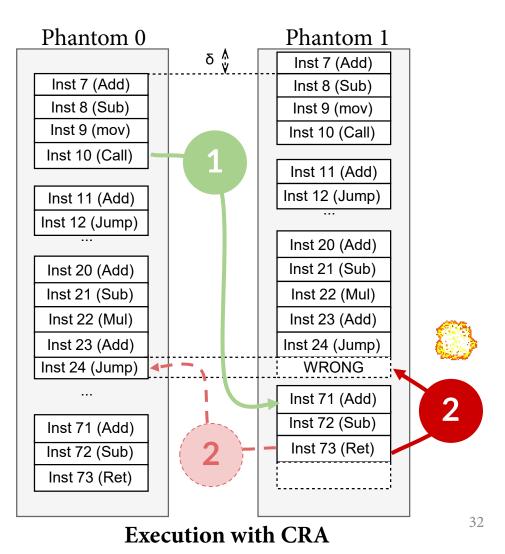
Phantoms force an adversary to guess the execution path.





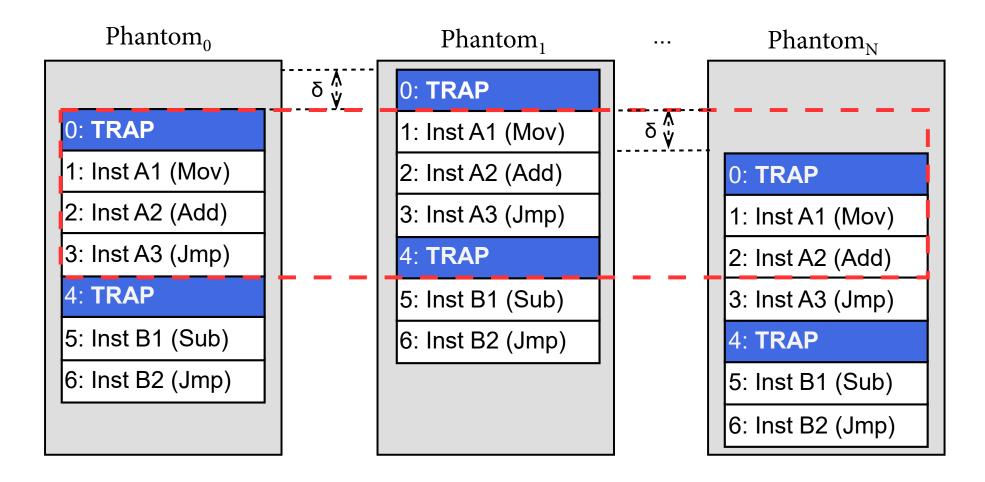
Phantoms force an adversary to guess the execution path.





How does PNS precisely trap an attacker?

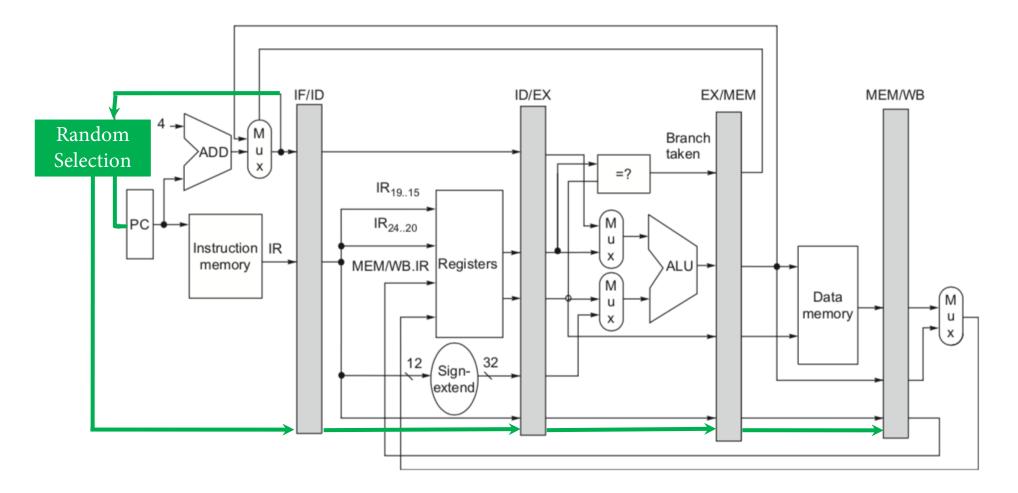
Code is instrumented with special instructions to throw an exception.



How is PNS implemented?

Ę

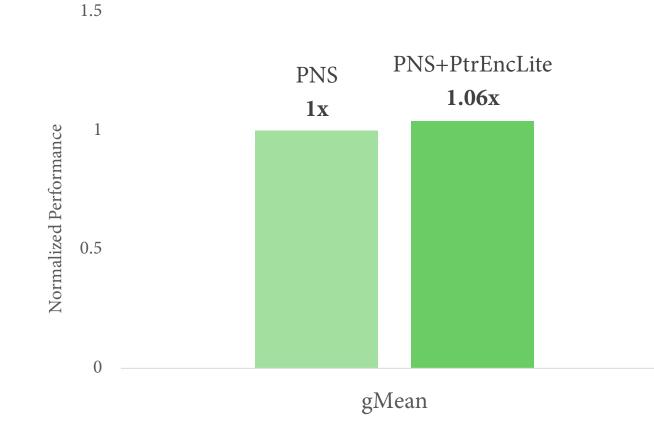
We do minimal changes to the processor frontend.



How was PNS evaluated?

F

We used the gem5 architectural simulator to validate correctness & performance.

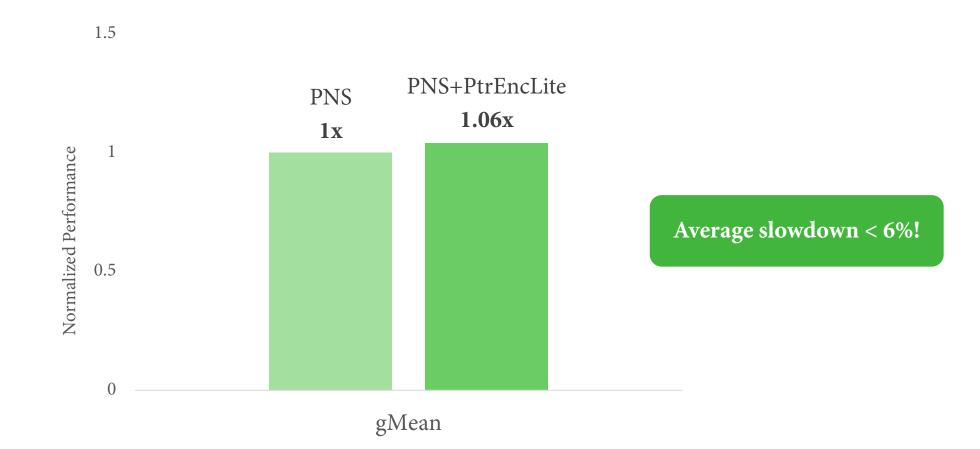


SPEC CPU2017

How was PNS evaluated?

Ę

We used the gem5 architectural simulator to validate correctness & performance.



SPEC CPU2017

Limitations



Repeated Observation Attack

Running the same binary on a non-protected system can leak the security shift of a return address.

Why is PNS well suited for constrained devices?

It brings efficient protection with minimal cost.



Minimal Performance Impact

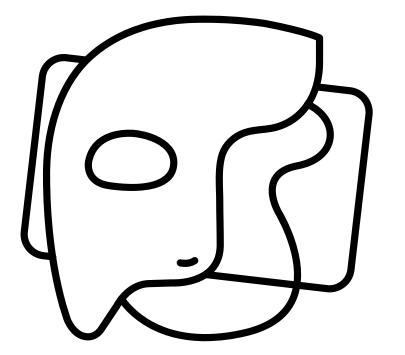
PNS has minimal impact on workload execution.



Memory Savings

PNS cuts down on resource duplication associated with aliasing multiple instructions.

Thanks for listening!



Phantom Name System

Find the paper here!



https://arxiv.org/pdf/1911.02038.pdf