Securing Resource Constrained Processors with Name Confusion

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

Columbia University
Brown University
09/21/2021
Embedded systems are everywhere!
Why focus on software threats?
CPSs are predominantly written in memory unsafe languages.

Source: Embedded Insights Survey 2019
Code Reuse Attacks

- Virtual Address Space
  - Instr 1: 0x4000
  - Instr 80: 0x5000
  - Instr 85: 0x5010

- Stack
  - Gadget₁
  - Gadget₂
  - Gadget₃
Traditional Architecture

An instruction has a single name (or address).
Traditional Architecture
An instruction has a single name (or address).

Information hiding techniques (e.g. ISR) randomize instruction encoding.
Traditional Architecture

An instruction has a single name (or address).

Metadata techniques (e.g. CFI) check instructions during execution.
Traditional Architecture

An instruction has a single name (or address).

Moving target techniques change instruction names over time.
Phantom Name System
A Name Confusion Architecture
Name Confusion Architecture

Multiple phantom addresses alias to an instruction.

Phantom Address Space

Virtual Address Space

0x4000

Instr 1

0x5000

Instr 1

0x6000

Instr 1

0x7000

Instr 1

Instr 1

0x4000
Name Confusion Architecture

Multiple phantom addresses alias to an instruction.

An attacker must guess which will be executed!
How are phantoms constructed?

Phantoms are logically displaced relative to the original program.
How are phantoms constructed?
Phantoms are logically displaced relative to the original program.

Phantom Address Space

Virtual Address Space

Physical Address Space

\[ \Delta \text{Domain Offset} \]

\[ \delta \text{Security Shift} \]
How are phantoms constructed?
Phantoms are logically displaced relative to the original program.
How does PNS diversify execution?
It diversifies the *path* of execution at every basic block.
How does PNS diversify execution?

It diversifies the *path* of execution at every basic block.
How does PNS diversify execution?

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

Program Control Flow Graph

Thwarts many Code-Reuse Attacks (CRAs).
## How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.

<table>
<thead>
<tr>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst 7 (Add)</td>
</tr>
<tr>
<td>Inst 8 (Sub)</td>
</tr>
<tr>
<td>Inst 9 (mov)</td>
</tr>
<tr>
<td>Inst 10 (Call)</td>
</tr>
<tr>
<td>Inst 11 (Add)</td>
</tr>
<tr>
<td>Inst 12 (Jump)</td>
</tr>
<tr>
<td>Inst 20 (Add)</td>
</tr>
<tr>
<td>Inst 21 (Sub)</td>
</tr>
<tr>
<td>Inst 22 (Mul)</td>
</tr>
<tr>
<td>Inst 23 (Add)</td>
</tr>
<tr>
<td>Inst 24 (Jump)</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Inst 71 (Add)</td>
</tr>
<tr>
<td>Inst 72 (Sub)</td>
</tr>
<tr>
<td>Inst 73 (Ret)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Normal Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.

<table>
<thead>
<tr>
<th>Inst 7 (Add)</th>
<th>Inst 8 (Sub)</th>
<th>Inst 9 (mov)</th>
<th>Inst 10 (Call)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst 11 (Add)</td>
<td>Inst 12 (Jump)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inst 20 (Add)</td>
<td>Inst 21 (Sub)</td>
<td>Inst 22 (Mul)</td>
<td>Inst 23 (Add)</td>
</tr>
<tr>
<td>Inst 24 (Jump)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inst 71 (Add)</td>
<td>Inst 72 (Sub)</td>
<td>Inst 73 (Ret)</td>
<td></td>
</tr>
</tbody>
</table>

Original

```
1  
```

Normal Execution
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?
Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.

Phantom 0

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)
Inst 73 (Ret)

Phantom 1

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)
Inst 73 (Ret)

Normal Execution
How does PNS protect against CRAs?
Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.

Phantom 0

1. 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)
   Inst 73 (Ret)

Phantom 1

1. 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)
   Inst 73 (Ret)

Normal Execution

Execution with CRA

Phantom 0

1. Inst 7 (Add)
   Inst 8 (Sub)
   Inst 9 (mov)
   Inst 10 (Call)
   ...
   Inst 20 (Add)
   Inst 21 (Sub)
   Inst 22 (Mul)
   Inst 23 (Add)
   Inst 24 (Jump)
   Inst 71 (Add)
   Inst 72 (Sub)
   Inst 73 (Ret)

Phantom 1

1. 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)
   Inst 73 (Ret)
How does PNS protect against CRAs?

Phantoms force an adversary to guess the execution path.
How does PNS protect against CRAs?

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?

We do minimal changes to the processor frontend.
How was PNS evaluated?
We used the gem5 architectural simulator to validate correctness & performance.

---

**SPEC CPU2017**
How was PNS evaluated?
We used the gem5 architectural simulator to validate correctness & performance.

Average slowdown < 6%!
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!

[Link to paper]