MERR: Improving Security of Persistent Memory Objects via Efficient Memory Exposure Reduction and Randomization

Yuanchao Xu, Yan Solihin, Xipeng Shen

NC STATE UNIVERSITY

UNIVERSITY OF CENTRAL FLORIDA
Persistent Memory (PM)

- Processor
- DRAM
- Disk

Id/st Instructions

Fast

Non-volatile

Persistent Data Structure
Security is more Important for PM

No System calls

Dangling Pointer  Victim Data

DRAM

Persistent Memory

Attacker

Users
Idea: Memory Exposure Reduction & Randomization

Program:

- `mmap (PM)`
- Access PM
- Access PM
- Access PM
- `munmap (PM)`

Memory Exposure

Time

Attacker
Idea: Memory Exposure Reduction & Randomization

Program:

- mmap (PM)
- access PM
- access PM
- access PM
- munmap (PM)

Time

Memory Exposure Reduction

- Attack Info.

Address Randomization

Attacker
MERR

Page Table Design

Architectural Support

MERR

70% Reduction 10% Overhead

80us Temporal Window

10^5 Randomization Frequency

Time
Outline

• Attach and Detach system calls
• Page table design
• Randomization
• Architecture support
• Evaluation
Attach & Detach

Attach (PMO ID, Permission)

Detach(PMO ID)

Process Address Space

Physical Address Space

PTEs

PMO

Persistent Memory Object (PMO)
Efficiency Challenge

- Attach (PMO ID, Permission)
- PMO Access

Process Hierarchy Page Table

- L3 Directory
- L2 Directory
- L1 Directory

Physical Address Space

- I GB
- PMO

262657 PTE Initiations!

Page Faults
Embedding Page Table Subtree

Process Hierarchy Page Table

Physical Address Space

L3 Directory

PTE

1GB PMO

Page Table Sub Tree

Metadata

Data

One PTE Modification!

Attach (PMO ID, Permission)

Detach (PMO ID)
PMO Space Layout Randomization

Process Hierarchy Page Table

L3 Directory
- PTE
- PTE
- PTE
- ...

Physical Address Space

1GB PMO
- Metadata
- Page Table Sub Tree
- Data
- Data

Attach (PMO ID, Permission)
Permission Control Challenges

Process Hierarchy Page Table

L3 Directory

PTE

Physical Address Space

1GB PMO

Metadata

Data

Page Table Sub Tree

Data

Attach (PMO ID, Permission)
Permission Control Challenges

Process Hierarchy Page Table

<table>
<thead>
<tr>
<th>L3 Directory</th>
<th>Physical Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach (PMO ID, RW)</td>
<td>1GB PMO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default Permission</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTE PTE PTE PTE</td>
<td>PTE PTE PTE PTE</td>
</tr>
</tbody>
</table>

Permission
Permission Control Challenges

- Attach (PMO ID, RW)
- Attach (PMO ID, RO)

Process 1 Hierarchy Page Table

L3 Directory

Process 2 Hierarchy Page Table

L3 Directory

Physical Address Space

1GB PMO

Different Permission

Data
Permission Matrix

1d/st instruction

Permission Matrix

Process-wide PMO Permission

Most Strict Permission

TLB

Page-level Permission

Attach

Detach
Permission Matrix Basic Design

Virtual Address (VA)

Permission Matrix

VA Start | VA End | Permission
--------|--------|------------
VA Start | VA End | Permission
VA Start | VA End | Permission
...

High Overhead in Critical Path!
Virtual Address Tag in Permission Matrix

Process Hierarchy Page Table

- **VA Prefix**
- **VA Suffix**

L2 Directory

- **PTE**
  - **VA Start**
  - **VA End**

Physical Address Space

- **PMO 4KB~2MB**

Page Table Subtree

- **PTE**
- **Invalid**

Data

- **Never Used**
Virtual Address Tag in Permission Matrix

VA Tag = VA Prefix + VA Suffix
Virtual Address Tag in Permission Matrix

VA Tag = VA Prefix + \[\begin{align*}
0 \text{ bit} & \quad \text{PMO Size} = 4\text{KB} \\
9\text{-bit } x...x & \quad 4\text{KB} < \text{PMO Size} \leq 2\text{MB} \\
18\text{-bit } x...x & \quad 2\text{MB} < \text{PMO Size} \leq 1\text{GB}
\end{align*}\]

Content Addressable Memory (CAM) Locate Exact one Entry From VA!

\[x\text{ means all match in Content Addressable Memory}\]
Permission Matrix Design

Virtual Address (VA)

Permission Matrix (CAM)

VA Tag → Permission
VA Tag → Permission
VA Tag → Permission

Process-wide PMO Permission
Evaluation Methodology

• **Workloads:**
  • WHISPER benchmarks

• **Operating System Overhead:**
  • Implement attach and detach library to replace mmap() and munmap()

• **Architectural Overhead:**
  • Intel Pin toolkit
  • Trace-Driven Simulation
Evaluation Metrics

• Attached Memory Exposure Rate (AMER)

\[ AMER = \frac{Reduced \ Time}{Original \ Time} \]
Evaluation Metrics

- Attached Memory Exposure Rate (AMER)
- Memory Exposure Window (MEW)
- PMO Space Layout Randomization Frequency (PSLR Frequency)

\[
\text{AMER} = \frac{\text{Reduced Time}}{\text{Original Time}}
\]

\[
\text{PSLR Frequency} = \frac{\text{Total Times}}{\text{Execution Time}}
\]

Time
Evaluation Metrics

• Attached Memory Exposure Rate (AMER)
• Memory Exposure Window (MEW)
• PMO Space Layout Randomization Frequency (PSLR Frequency)

\[
\text{AMER} = \frac{\text{Reduced Time}}{\text{Original Time}}
\]

\[
\text{MEW}
\]

\[
\text{PSLR Frequency} = \frac{\text{Execution Time}}{3}
\]
Performance

• About 10% overhead
  - 30% AMER (70% Reduction)
  - 80us Maximal MEW
  - PSLR per 41us

• To achieve above goals

Enhanced ASLR

2700x Speedup

More results in the paper

Basic Attach and Detach

Overhead (%) vs. Size of PMO (MB)

10^5 per Second Frequency
Security of MERR

- **No Protection**: 
  - ~10ms
  - Probe, Weaponize, Attack

- **Onetime ASLR**: 
  - ~seconds
  - Probe, Weaponize, Attack

- **80us Maximal MEW**: 
  - Error, Probe, Error, Probe, Error

- **+ PSLR**: 
  - Error, Probe, Error, Probe, Error

**More analysis in the paper**

80us Attacker Surface
Conclusion

• New angle to improve security through reduce memory exposure and randomization

• Improved efficiency by page table design and architectural support

• Achieved 70% memory exposure reduction and 80us memory exposure window with about 10% overhead on real world applications

• Provided An Order-of-magnitude speedup compared to state-of-art runtime randomization