Review “LMP”

Name: Wenhui Zhang

Paper Name：

Huang, W., Huang, Z., Miyani, D., & Lie, D. (2016, December). LMP: light-weighted memory protection with hardware assistance. In *Proceedings of the 32nd Annual Conference on Computer Security Applications* (pp. 460-470). ACM.

Contribution:

This paper proposed a method to protect return boundaries by using a Shadow Stack. It use MPX instructions to perform validation and protection of shadow stack.

Motivation:

- Prevent memory corruption – ROP attacks
- Existing CFI techniques are expensive and incur significant overheads (~ 2x)
- Leverage Intel MPX hardware extensions to improve CFI performance

Related works:

Intel MPX

Methodology:

Lifetime of Shadow Stack is as: (1) Reserve a specific region of the address space for shadow stacks; (2) Allocate multiple shadow stacks contiguously for multi-threaded applications; (3) Dynamically allocate and map stack space from the reserved area when creating threads and their shadow stacks; (4) Use mapping table to store offset of a thread’s shadow stack from its actual call stack; (5) Partition the reserved area into fixed size shadow stacks and use a table to indicate which ones are active or free; (6) On thread creation, assign an unallocated shadow stack and update mapping table with the offset; (7) On thread destruction, deallocate its shadow stack and clear the offset in mapping table.

On function entry: (1)Prepare shadow stack address in **%rax ; (2)** Copy return address at **%rsp** to shadow stack

On function exit: (1) Copy return address from shadow stack to MPX bound register **%bnd0; (2)** Use bound checking instructions to compare return address at **%rsp** and **%bnd0**

Shadow stack address is found by indexing into the stack region using the thread’s TID. The return address from the shadow stack is set as both upper and lower bound in the bound register **%bnd0.** Using **BNDCU** and **BNDCL** to compare instead of series of cmp/jmp instructions improves checking performance

Results:

Execution time has 3.9% avg. overhead, with 12.55% max for h264ref (has significantly more function calls/rets) and 2.12% avg. overhead excluding h264ref. Overhead components includes: Context settings: 0.1%; Bounds checking: 0.52%; Shadow stack ops: 3.27%; 84% of overheads is from shadow stack ops; 16% is from the memory protection system

Take away:

Special registers (BND0-BND3) in MPX is used for memory privilege segmentation. And we could conduct checks for security on privileged segments.