Off-by-one Overflow

- Overflow a buffer, but only by one byte

Off-by-one Overflow

- Overflow a buffer, but only by one byte
- Realistic!

Off-by-one Overflow

- Overflow a buffer, but only by one byte
- Realistic!
  - String terminator makes buffer exceed boundary

Off-by-one Overflow

- Overflow a buffer, but only by one byte
- Realistic!
  - String terminator makes buffer exceed boundary
  - while (i <= max) ...

Off-by-one Overflow

- Overflow a buffer, but only by one byte
- Realistic!
  - String terminator makes buffer exceed boundary
  - while (i <= max) ...
  - for (i = 0; i <= max; i++) ...

The stack and related registers

Two important registers:

- %ebp (Frame Pointer)

The stack and related registers

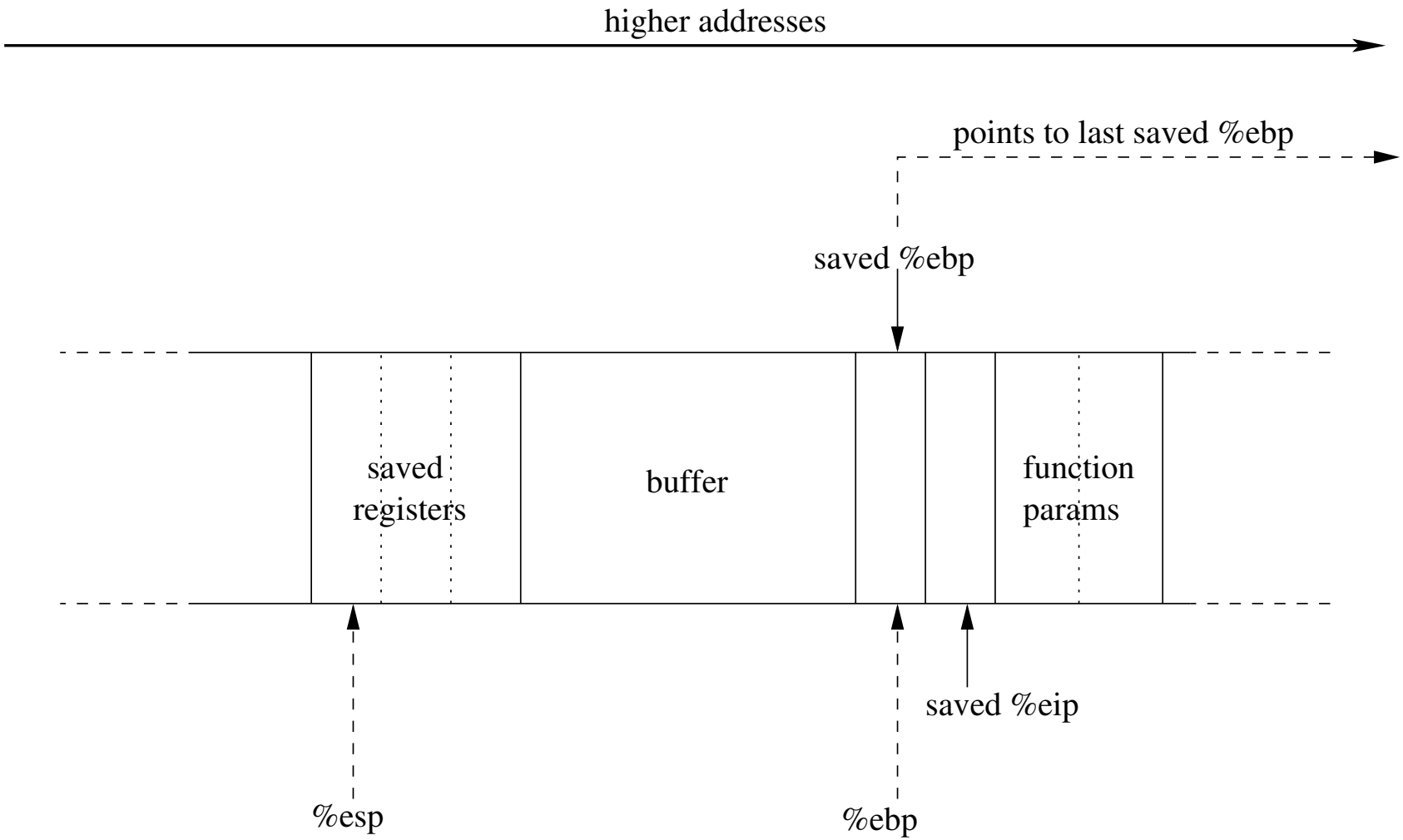
Two important registers:

- %ebp (Frame Pointer)
- %esp (Stack Pointer)

The stack and related registers

Two important registers:

- %ebp (Frame Pointer)
- %esp (Stack Pointer)



Overflow

- Little Endian Architecture

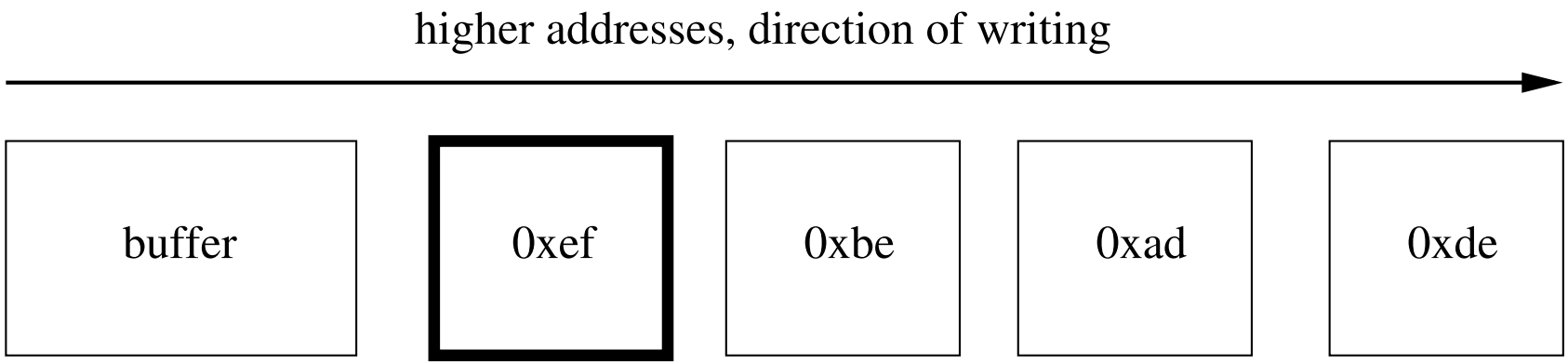


Figure: %ebp overwrite in detail

Push and Pop

- Putting data on the stack, and getting it back

Push and Pop

- Putting data on the stack, and getting it back
- Uses stack pointer %esp for operations

Push and Pop

- Putting data on the stack, and getting it back
- Uses stack pointer %esp for operations
- Push decrements %esp by 4 and stores value where %esp points to

Push and Pop

- Putting data on the stack, and getting it back
- Uses stack pointer %esp for operations
- Push decrements %esp by 4 and stores value where %esp points to
- Pop fetches value from where %esp points to and increments %esp then by 4

Leave

- Opcode used when leaving the frame

Leave

- Opcode used when leaving the frame
  - mov esp, ebp (%esp is set to %ebp)

Leave

- Opcode used when leaving the frame
  - mov esp, ebp (%esp is set to %ebp)
  - pop ebp (ebp is set to the saved ebp (!))

Ret

- Opcode to make a function return

Ret

- Opcode to make a function return
  - pop eip (eip is fetched from where esp points to (!))

Putting it together

- Overflow by one byte $\longrightarrow$ Old %ebp partly controlled by us

Putting it together

- Overflow by one byte $\rightarrow$ Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) $\rightarrow$ Not harmful

Putting it together

- Overflow by one byte $\rightarrow$ Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) $\rightarrow$ Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)

Putting it together

- Overflow by one byte $\rightarrow$ Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) $\rightarrow$ Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)
  - **Current** %ebp partly controlled by us

Putting it together

- Overflow by one byte → Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) → Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)
  - **Current** %ebp partly controlled by us
- Ret (old %eip popped from stack) → Not harmful

Putting it together

- Overflow by one byte → Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) → Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)
  - **Current** %ebp partly controlled by us
- Ret (old %eip popped from stack) → Not harmful
- Now back in caller function frame

Putting it together

- Overflow by one byte $\rightarrow$ Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) $\rightarrow$ Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)
  - **Current** %ebp partly controlled by us
- Ret (old %eip popped from stack) $\rightarrow$ Not harmful
- Now back in caller function frame
- Leave Step 1 (%esp set to %ebp)

Putting it together

- Overflow by one byte $\rightarrow$ Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) $\rightarrow$ Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)
  - **Current** %ebp partly controlled by us
- Ret (old %eip popped from stack) $\rightarrow$ Not harmful
- Now back in caller function frame
- Leave Step 1 (%esp set to %ebp)
  - **Current** %esp partly controlled by us

Putting it together

- Overflow by one byte $\rightarrow$ Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) $\rightarrow$ Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)
  - **Current** %ebp partly controlled by us
- Ret (old %eip popped from stack) $\rightarrow$ Not harmful
- Now back in caller function frame
- Leave Step 1 (%esp set to %ebp)
  - **Current** %esp partly controlled by us
- Leave Step 2 (old %ebp popped back to %ebp)

Putting it together

- Overflow by one byte → Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) → Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)
  - **Current** %ebp partly controlled by us
- Ret (old %eip popped from stack) → Not harmful
- Now back in caller function frame
- Leave Step 1 (%esp set to %ebp)
  - **Current** %esp partly controlled by us
- Leave Step 2 (old %ebp popped back to %ebp)
- Ret (old %eip popped from stack)

Putting it together

- Overflow by one byte $\rightarrow$ Old %ebp partly controlled by us
- Leave Step 1 (%esp set to %ebp) $\rightarrow$ Not harmful
- Leave Step 2 (old %ebp popped back to %ebp)
  - **Current** %ebp partly controlled by us
- Ret (old %eip popped from stack) $\rightarrow$ Not harmful
- Now back in caller function frame
- Leave Step 1 (%esp set to %ebp)
  - **Current** %esp partly controlled by us
- Leave Step 2 (old %ebp popped back to %ebp)
- Ret (old %eip popped from stack)
  - We partly control where %eip is popped from