Remix: On-demand Live Randomization

Yue Chen, Zhi Wang, David Whalley, Long Lu*

Florida State University, Stony Brook University*

Background

- Buffer Overflow -> Code Injection Attack

Background

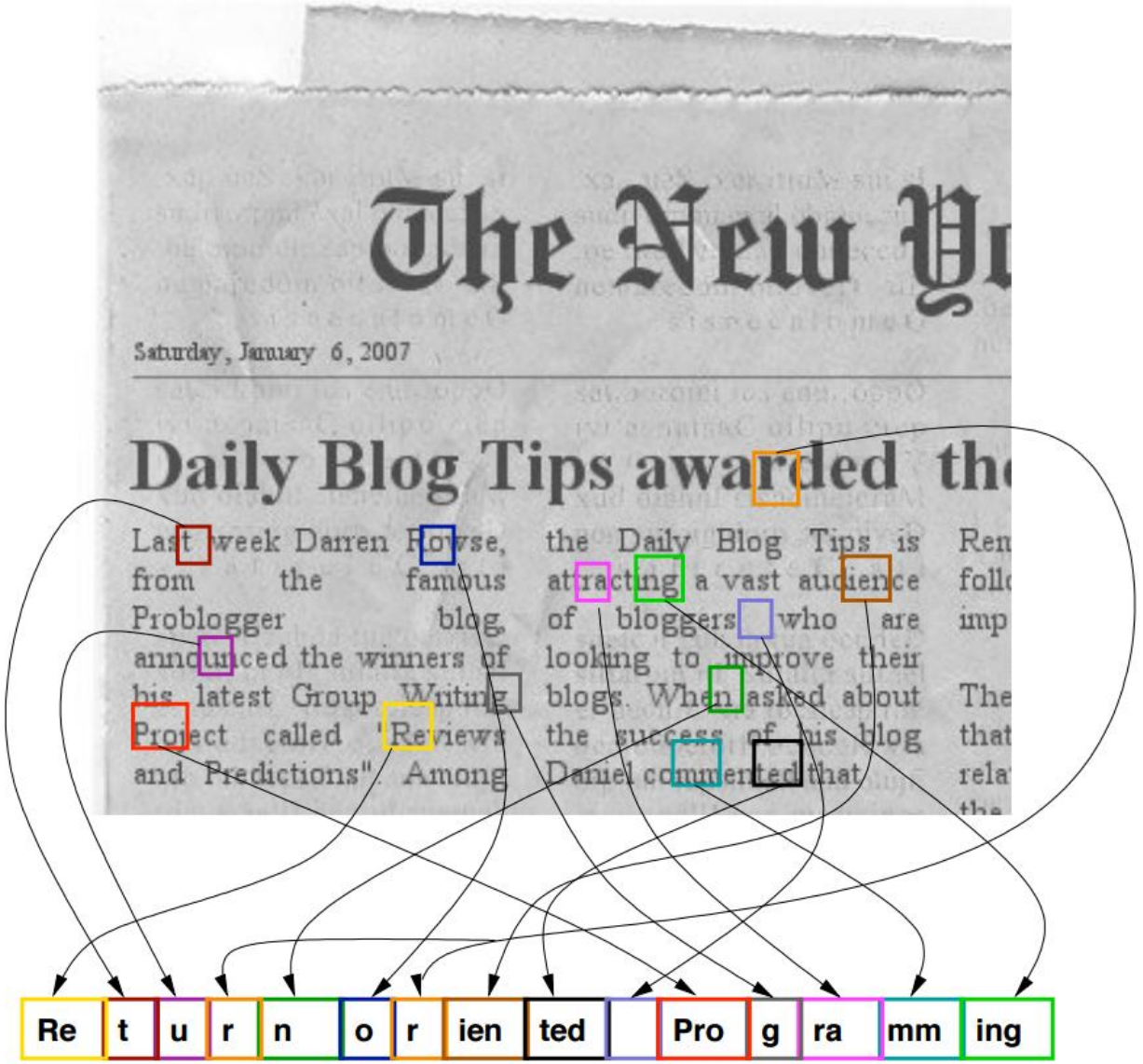
- Buffer Overflow -> Code Injection Attack
  - Defense: Data Execution Prevention (DEP)
    - Write XOR Execute: a block (page) of memory cannot be marked as both writable and executable at the same time.

Background

- Buffer Overflow -> Code Injection Attack
  - Defense: Data Execution Prevention (DEP)
    - Write XOR Execute: a block (page) of memory cannot be marked as both writable and executable at the same time.
- Return-oriented Programming (ROP) Attack
  - Discover gadgets from **existing** code, and chain them by **ret** instructions
  - Can be Turing complete

Code Reuse Attack

Existing Code



Chained Gadgets

ROP Defense Strategy

- ROP is one example of a broad class of attacks that require attackers to know or predict the locations of binary features.

ROP Defense Strategy

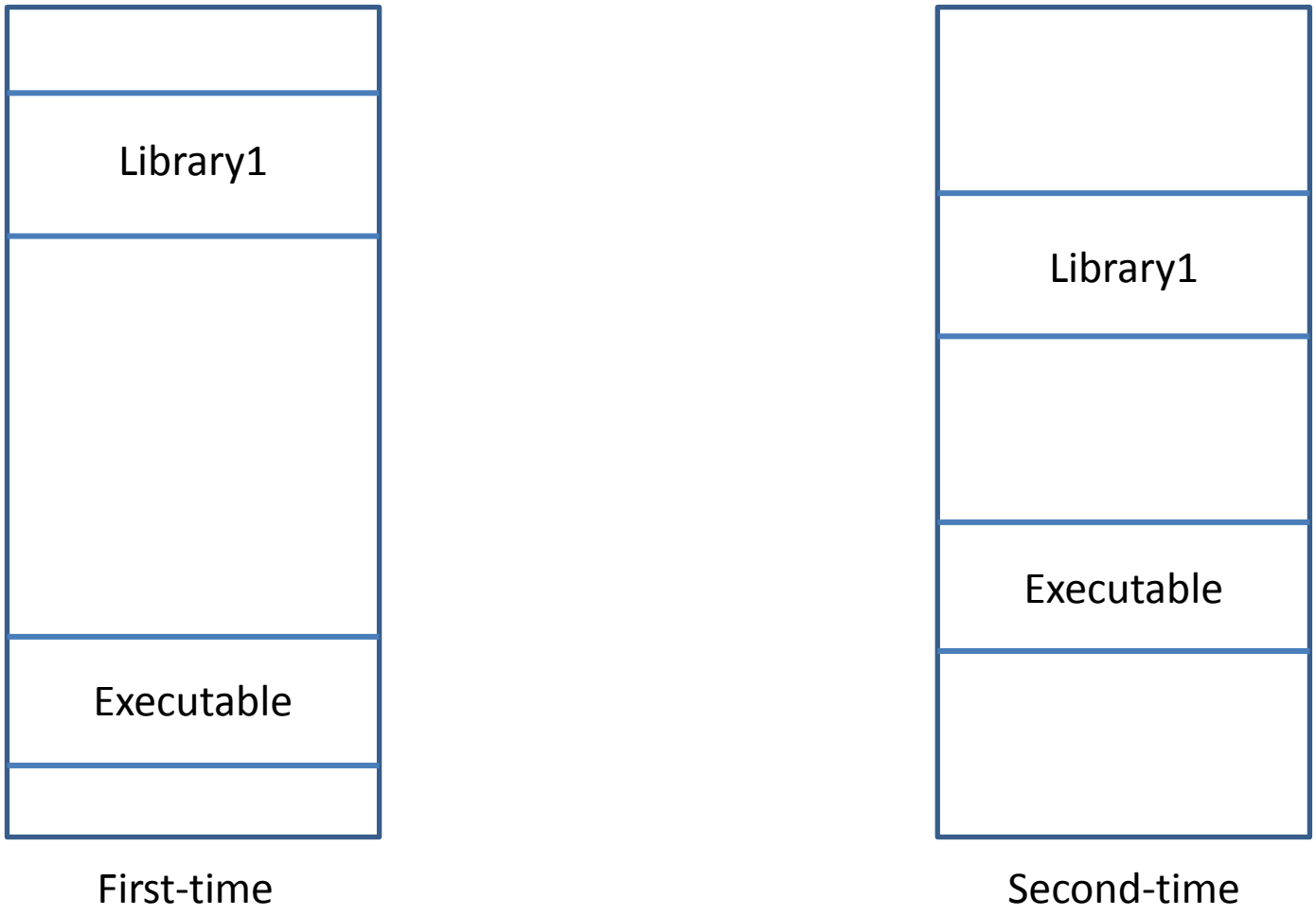
- ROP is one example of a broad class of attacks that require attackers to know or predict the locations of binary features.

Defense Goal

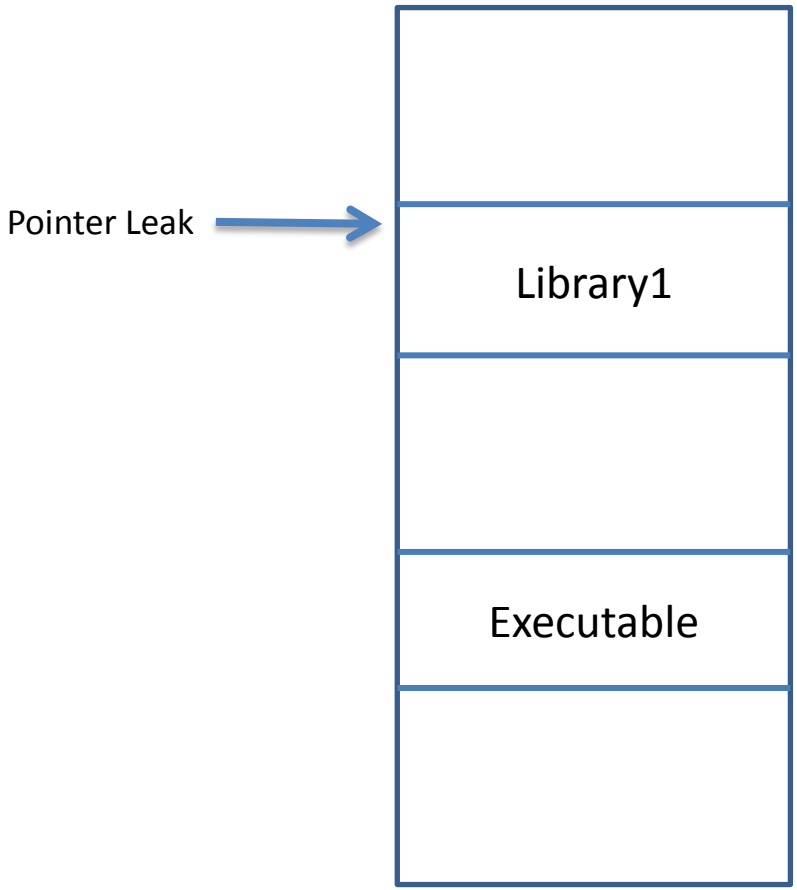
Frustrate such attacks by randomizing feature space, or remove features

ASLR

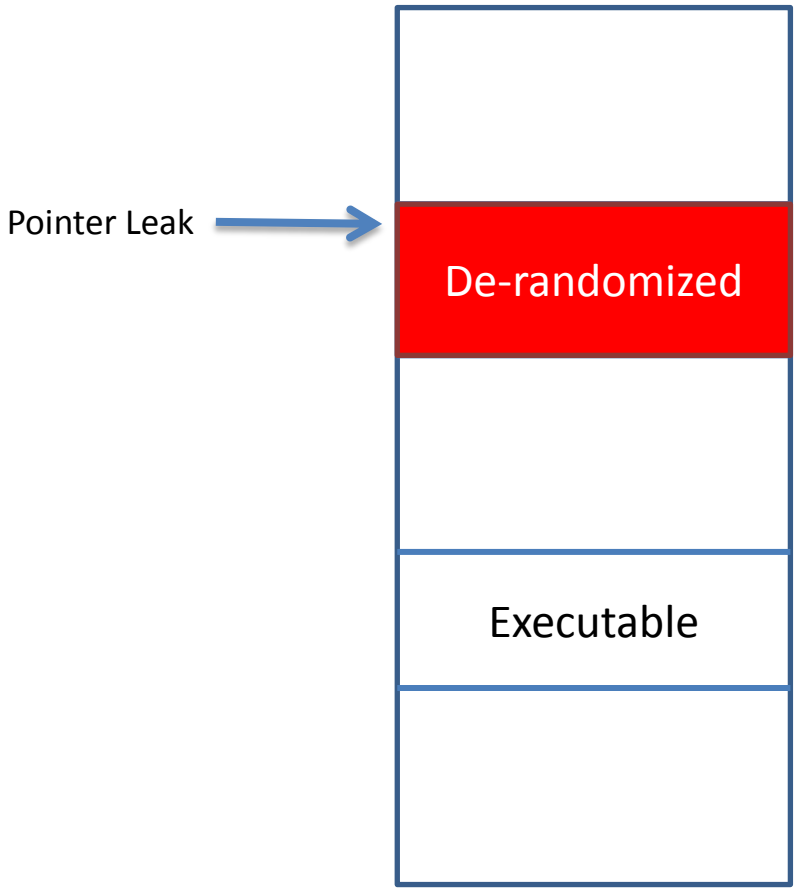
Address **S**pace **L**ayout **R**andomization



ASLR - Problem



ASLR - Problem



ASLR - Problem

Brute force attacks are still possible (When the entropy is small. E.g., 32-bit systems.)

Randomized only once.

Goal

- Live randomization during runtime
- Finer-grained randomization unit
- Low performance overhead
- Highly composable
  - Can and should be combined with other defenses (traditional ASLR, function randomization, etc.)

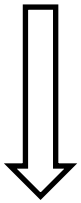
Remix

Live basic block (BB) (re-)randomization
within functions

A **basic block** is a straight-line code without jumps in or out of the middle of the block.

Remix

Live basic block (BB) (re-)randomization
within functions

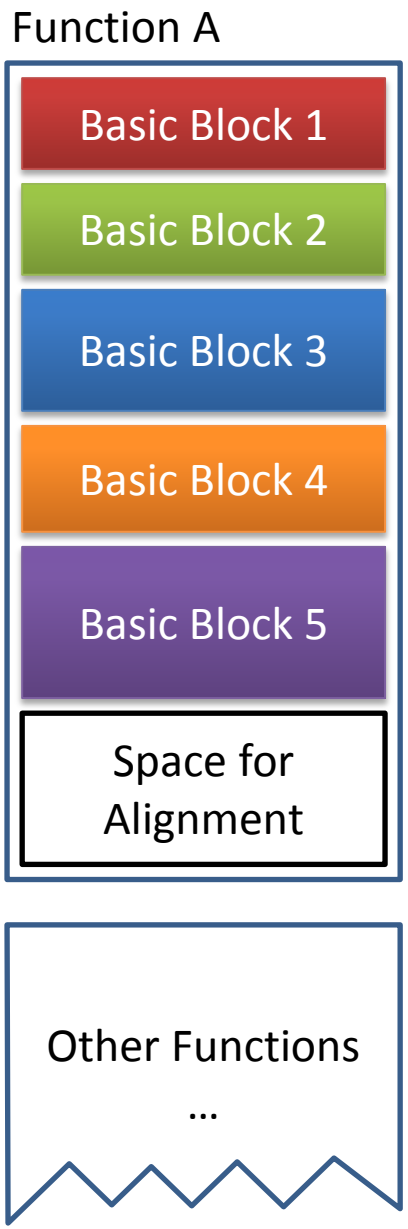


Advantages:

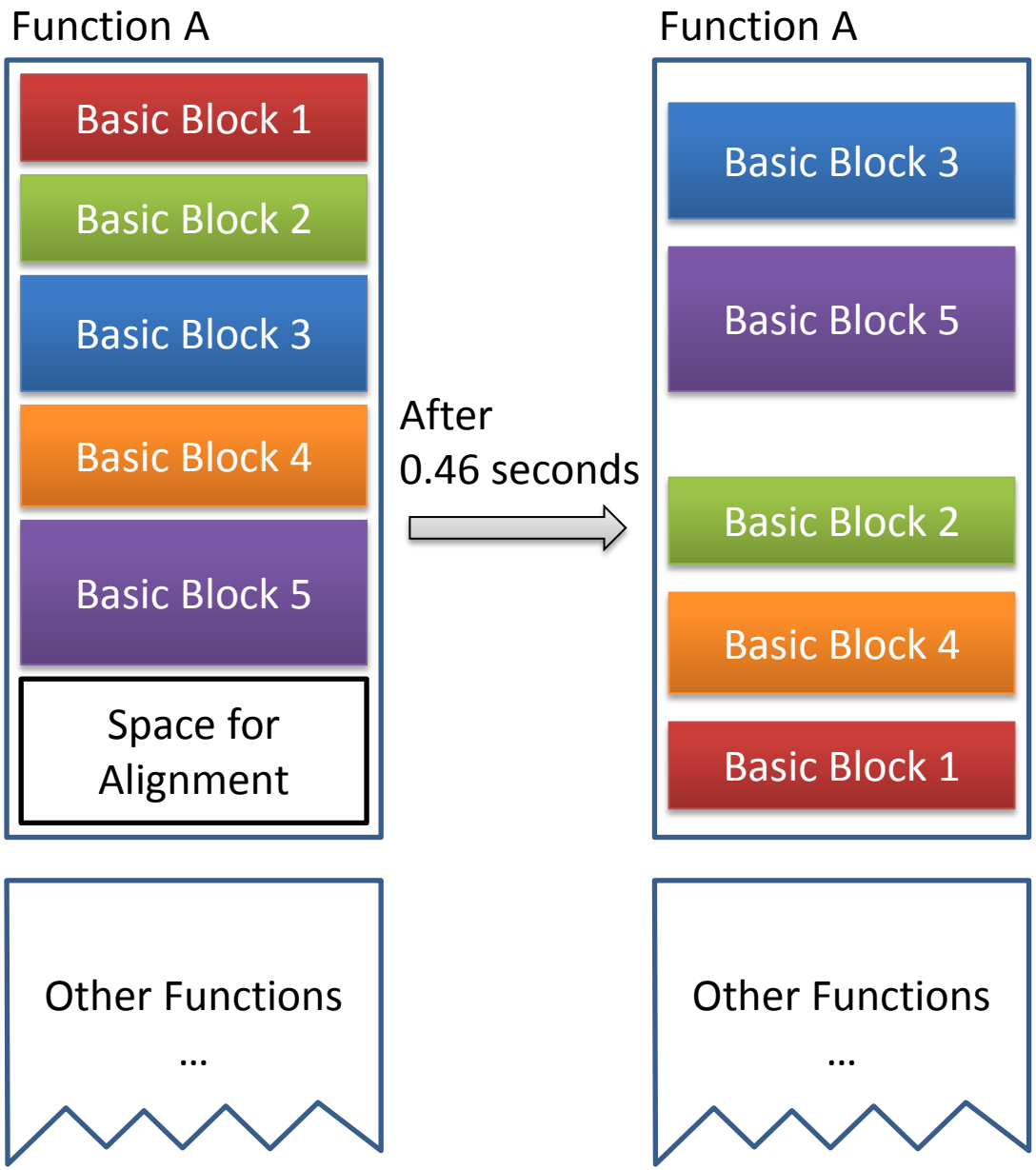
- No function pointer migration
- Good spatial locality

A **basic block** is a straight-line code without jumps in or out of the middle of the block.

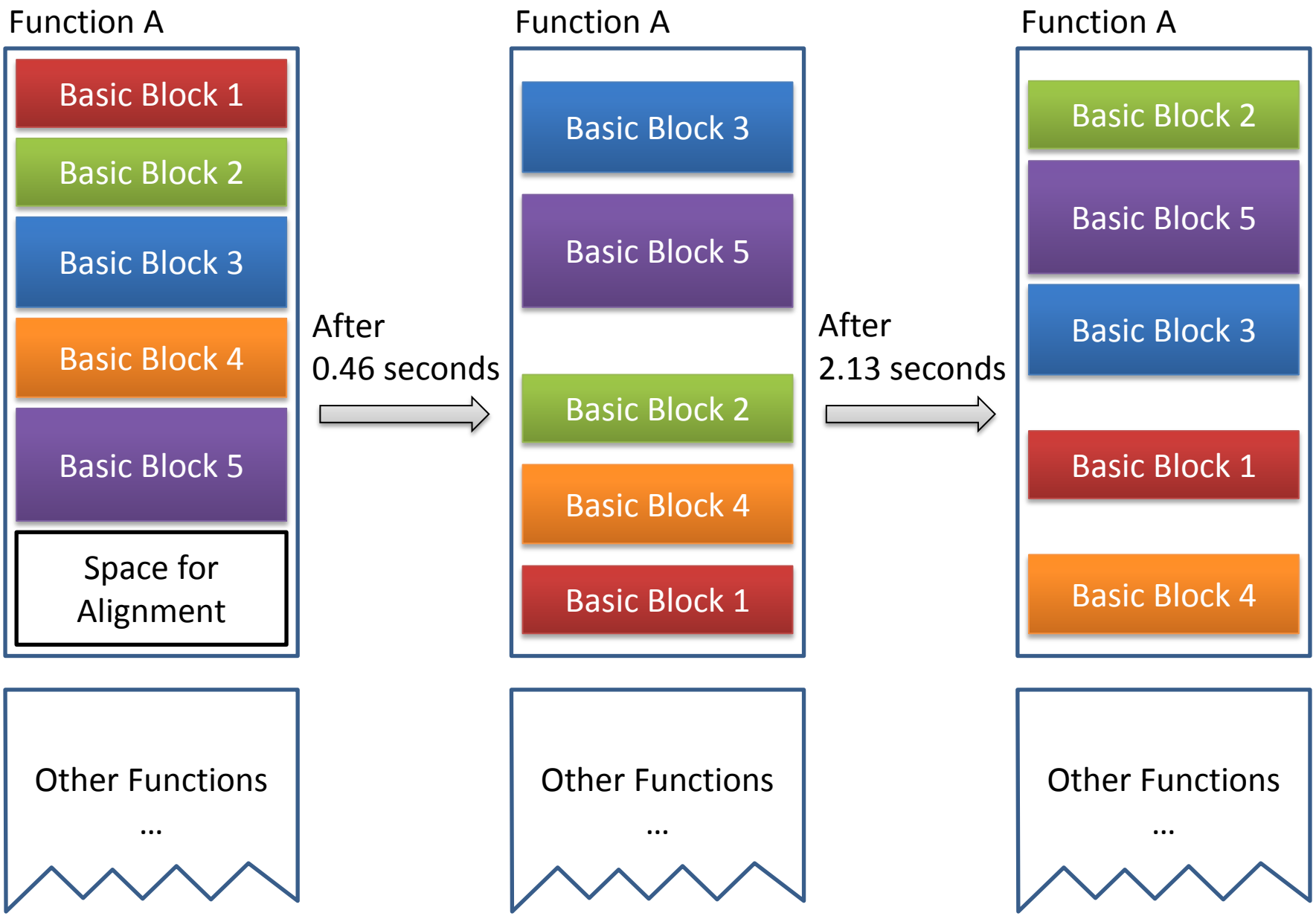
Remix


Function A
Basic Block 1
Basic Block 2
Basic Block 3
Basic Block 4
Basic Block 5
Space for Alignment
Other Functions
…

Remix


Function A
Basic Block 1
Basic Block 2
Basic Block 3
Basic Block 4
Basic Block 5
Space for Alignment
Other Functions
...
After
0.46 seconds
Function A
Basic Block 3
Basic Block 5
Basic Block 2
Basic Block 4
Basic Block 1
Other Functions
...

Remix

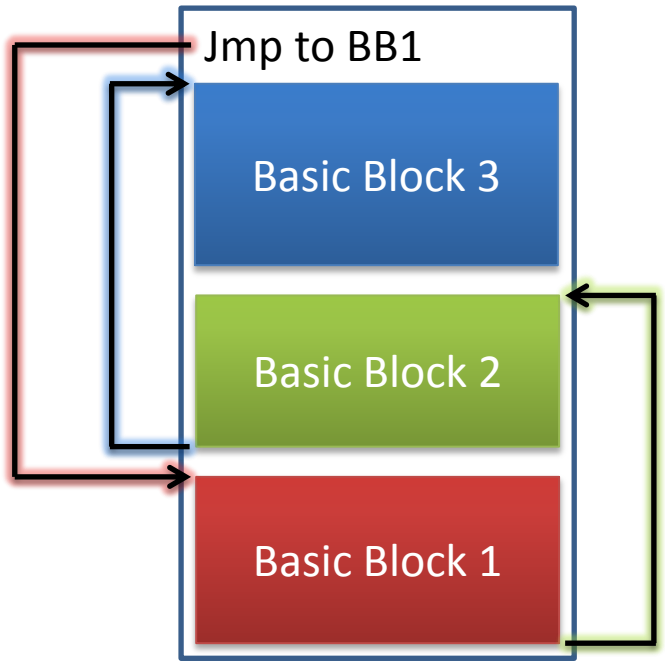

Function A
Basic Block 1
Basic Block 2
Basic Block 3
Basic Block 4
Basic Block 5
Space for Alignment
Other Functions
...
After
0.46 seconds
Function A
Basic Block 3
Basic Block 5
Basic Block 2
Basic Block 4
Basic Block 1
Other Functions
...
After
2.13 seconds
Function A
Basic Block 2
Basic Block 5
Basic Block 3
Basic Block 1
Basic Block 4
Other Functions
...

Challenges

- Chain randomized basic blocks together
  - Need extra space to chain basic blocks
  - Need to update instructions
- Stale pointer migration

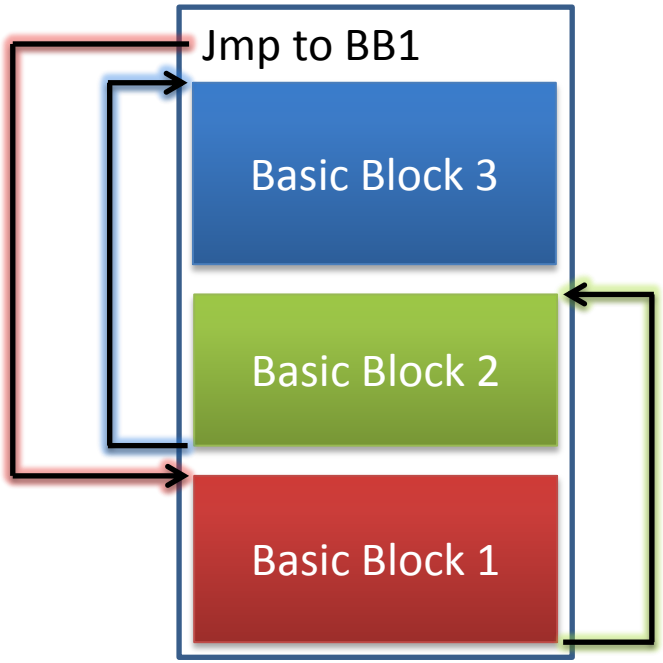
Extra Space
Case 1: Extra Jmp

(1) At the beginning of a function



Extra Space
Case 1: Extra Jmp

(1) At the beginning of a function

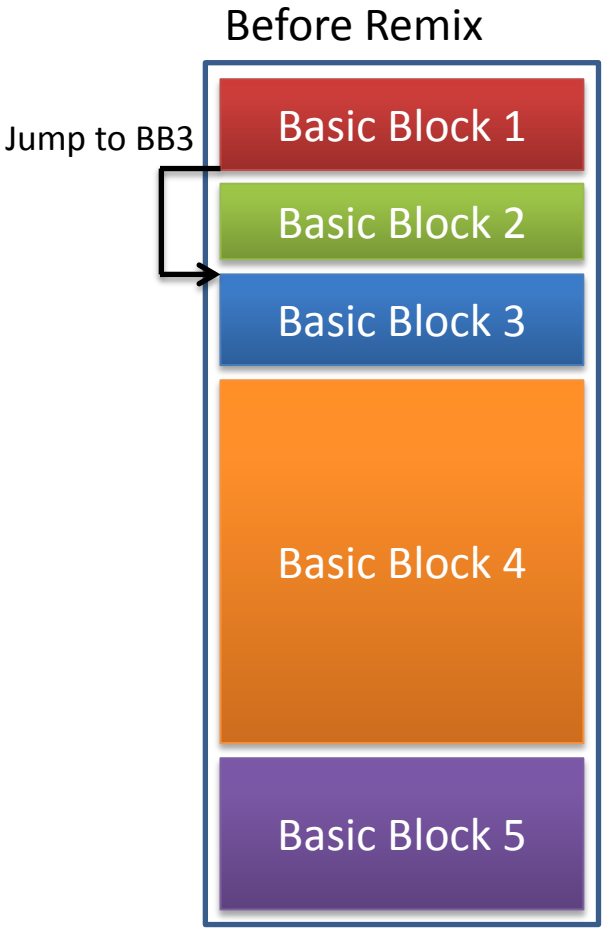


(2) At the end of a basic block that does not end with an instruction like **jmp/ret**

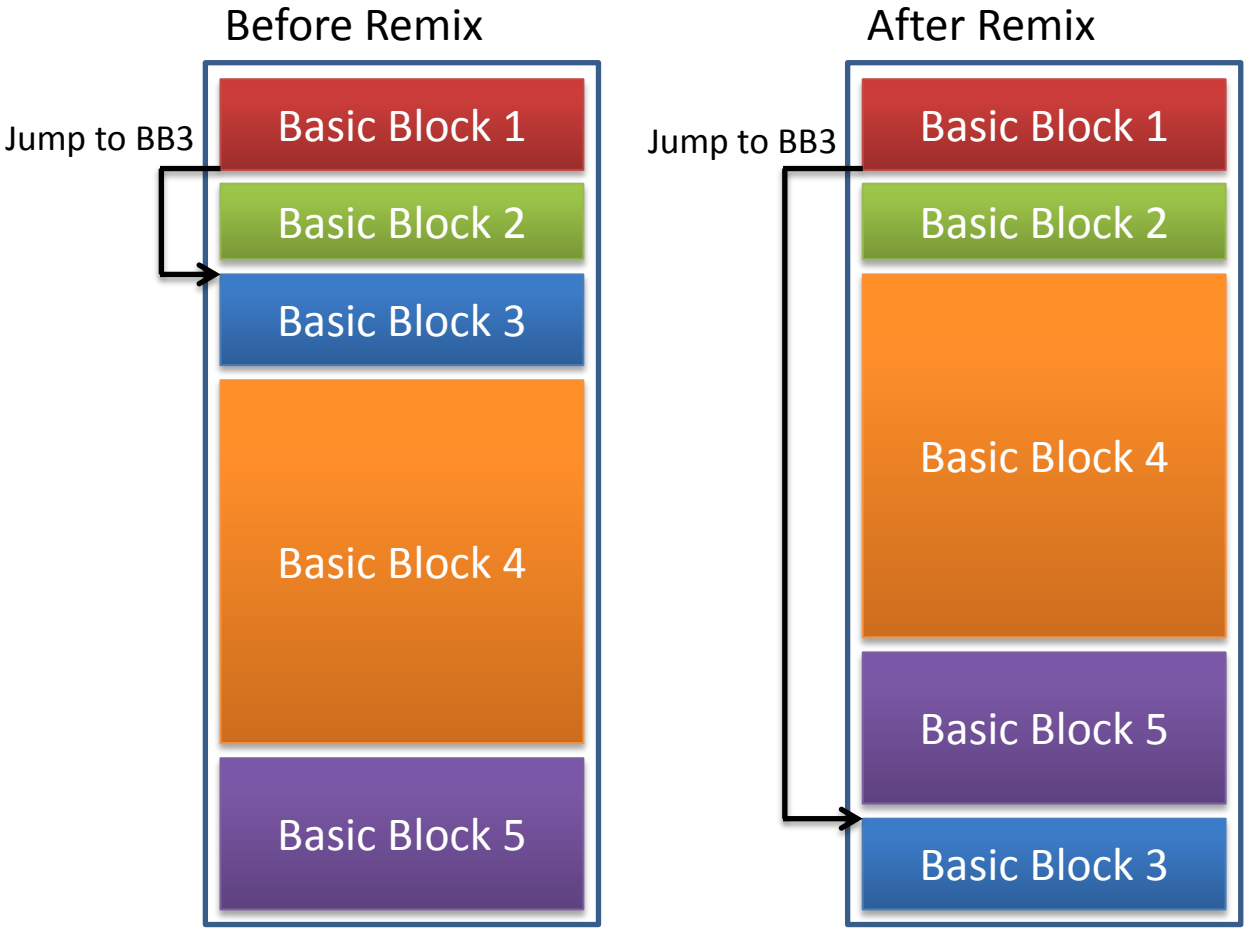
```
mov   ...
add   ...
mov   ...
```

```
mov   ...
add   ...
jle   ...
```

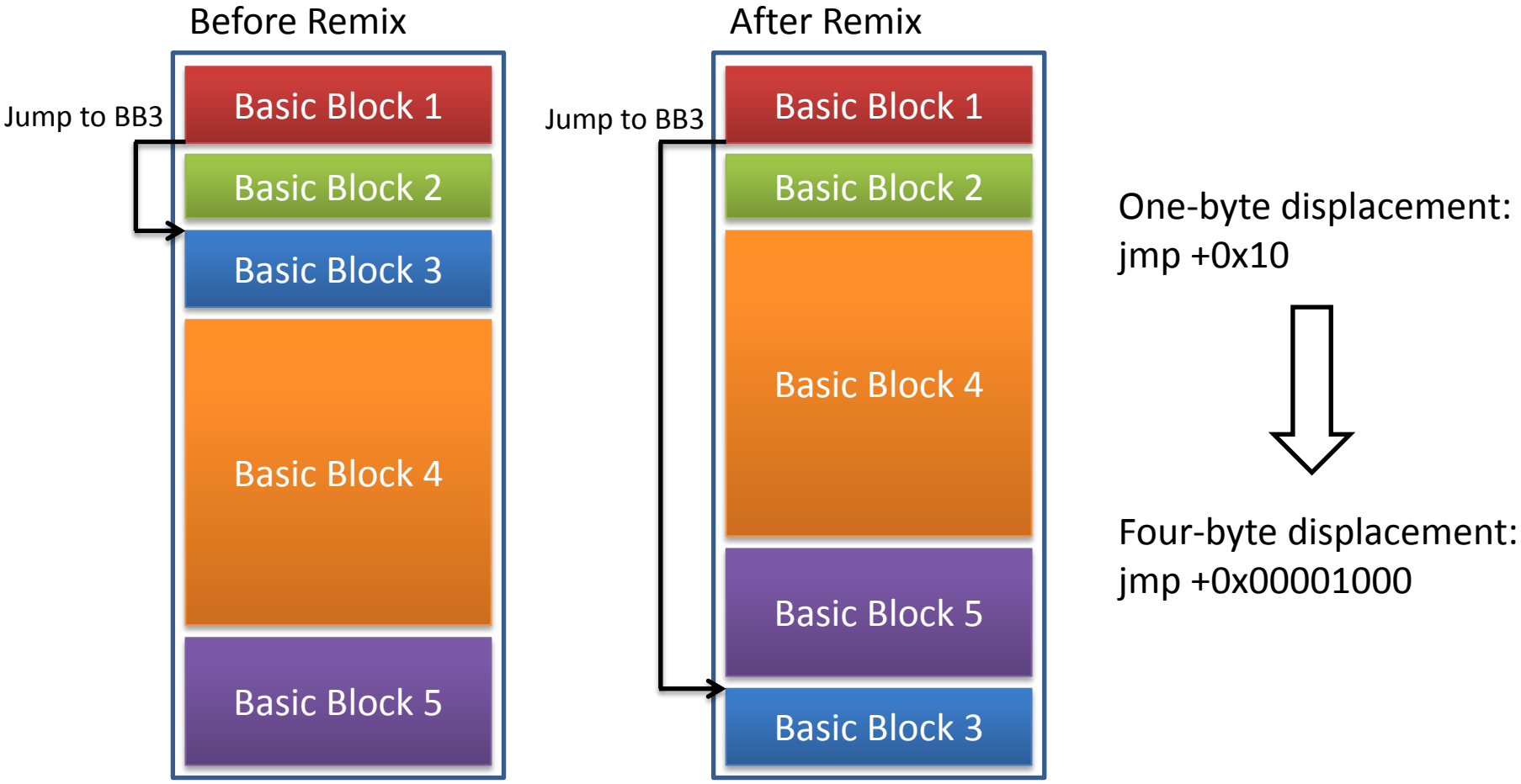
Extra Space
Case 2: Larger Displacement



Extra Space
Case 2: Larger Displacement



Extra Space
Case 2: Larger Displacement



Extra Space Solution

With Source Code:

Modify the compiler to:

1. Insert an extra 5-byte NOP instruction after each basic block
2. Only generate instructions with 4-byte displacement

✓ Enough Space Guaranteed!

Extra Space Solution

With Source Code:

Modify the compiler to:

1. Insert an extra 5-byte NOP instruction after each basic block
2. Only generate instructions with 4-byte displacement

✓ Enough Space Guaranteed!

Without Source Code:

Leverage existing NOP paddings:

- Function alignment
- Loop alignment

Can insert **random bytes** into NOP space after randomization, making attack even harder.

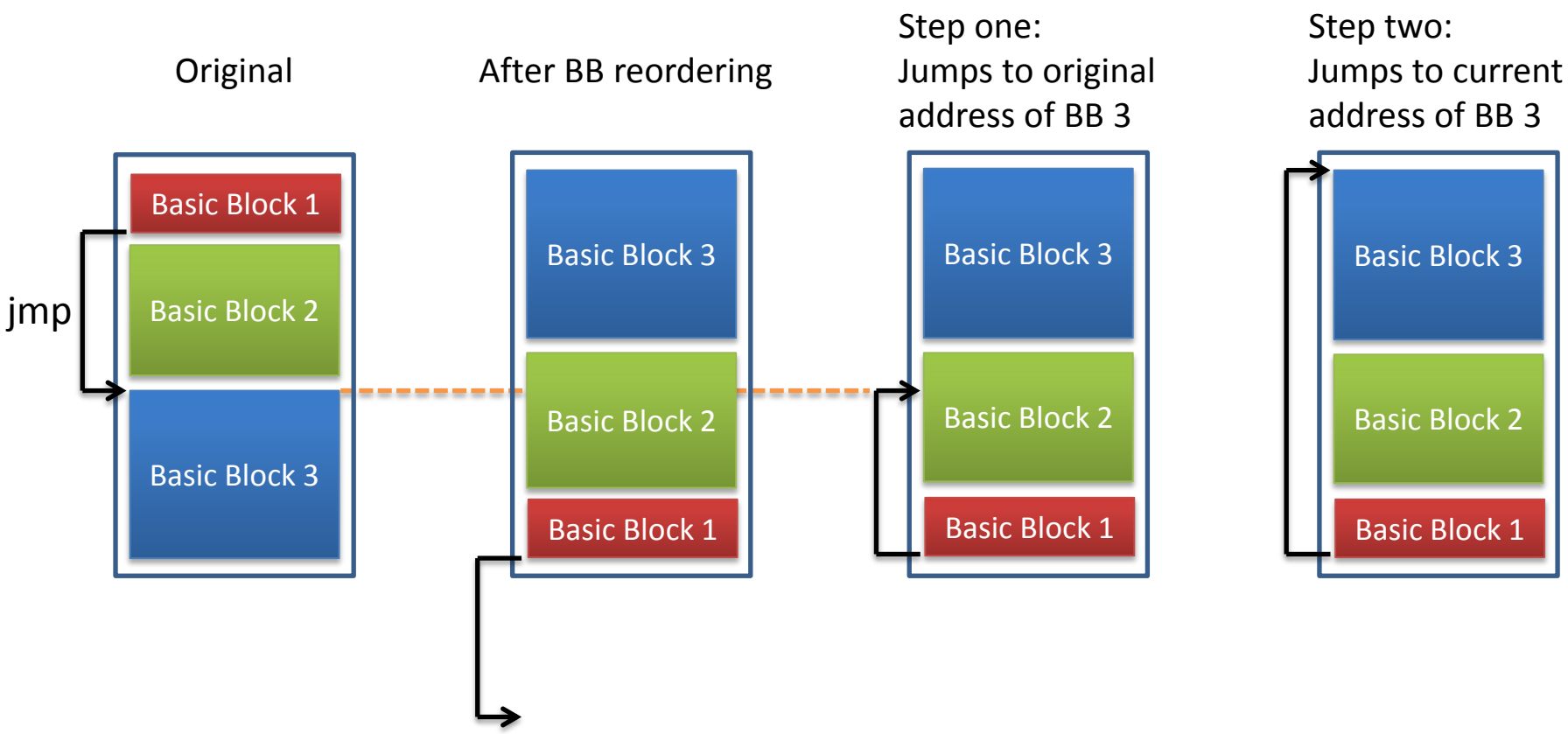
Instruction Update

Q: Which instructions need updating ?

A: **Control-flow** related ones, to adjust displacement

Instruction Update

Two-step update (e.g., unconditional direct jmp):



Instruction Update

- **Direct call**: step-one update
- **Indirect call**: no update needed
- **Direct jump**: step-one and step-two update
- **Indirect jump**: discussed later
- **PC-relative addressing**: step-one update

Indirect Jump

- Jump to functions – unchanged
  - PLT/GOT
  - Tail/Sibling Call
- Jump to basic blocks – see next

Basic Block Pointer Conversion

- **Why?**
  - Migrate stale pointers to basic blocks, to ensure correctness

Basic Block Pointer Conversion

- **Why?**
  - Migrate stale pointers to basic blocks, to ensure correctness
- **Where?**
  - Return address
  - Jump table (switch/case)
  - Saved context (e.g., setjmp/longjmp)
  - Kernel exception table

  …

Illustration - Return Address

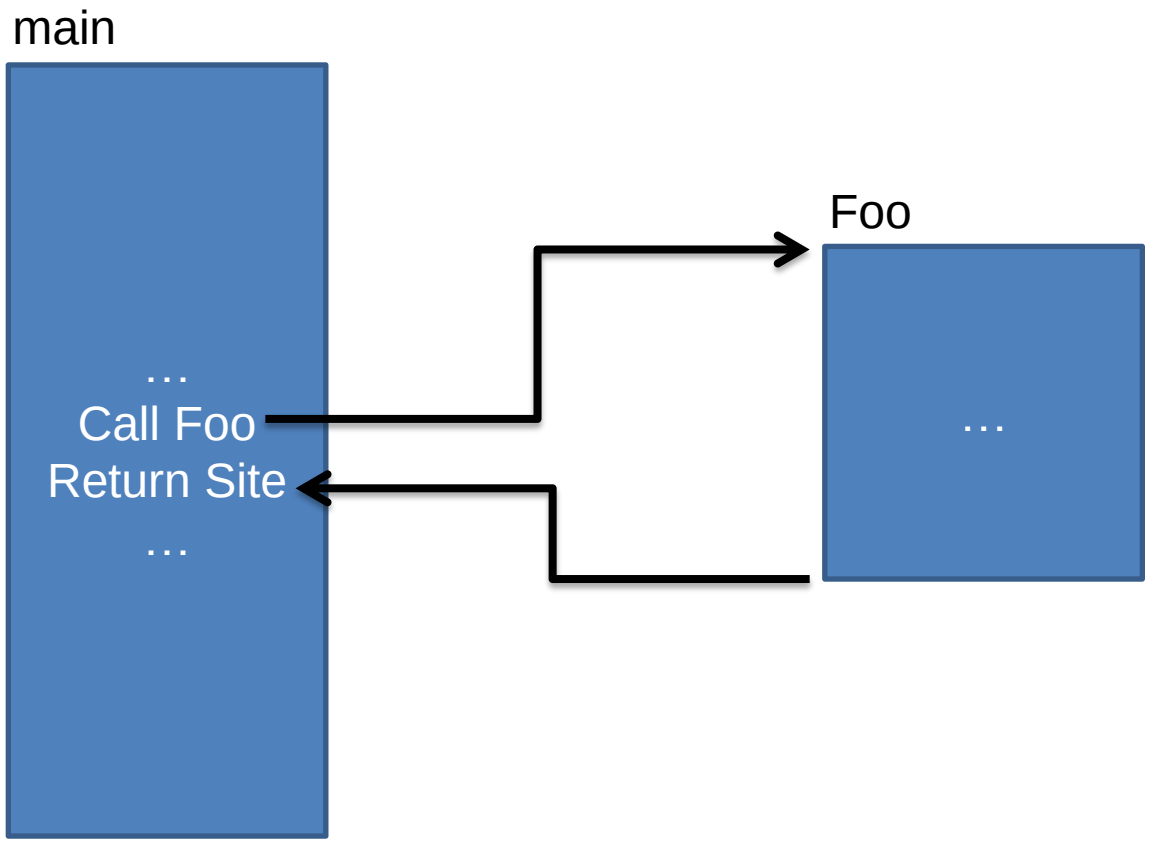


Illustration - Return Address

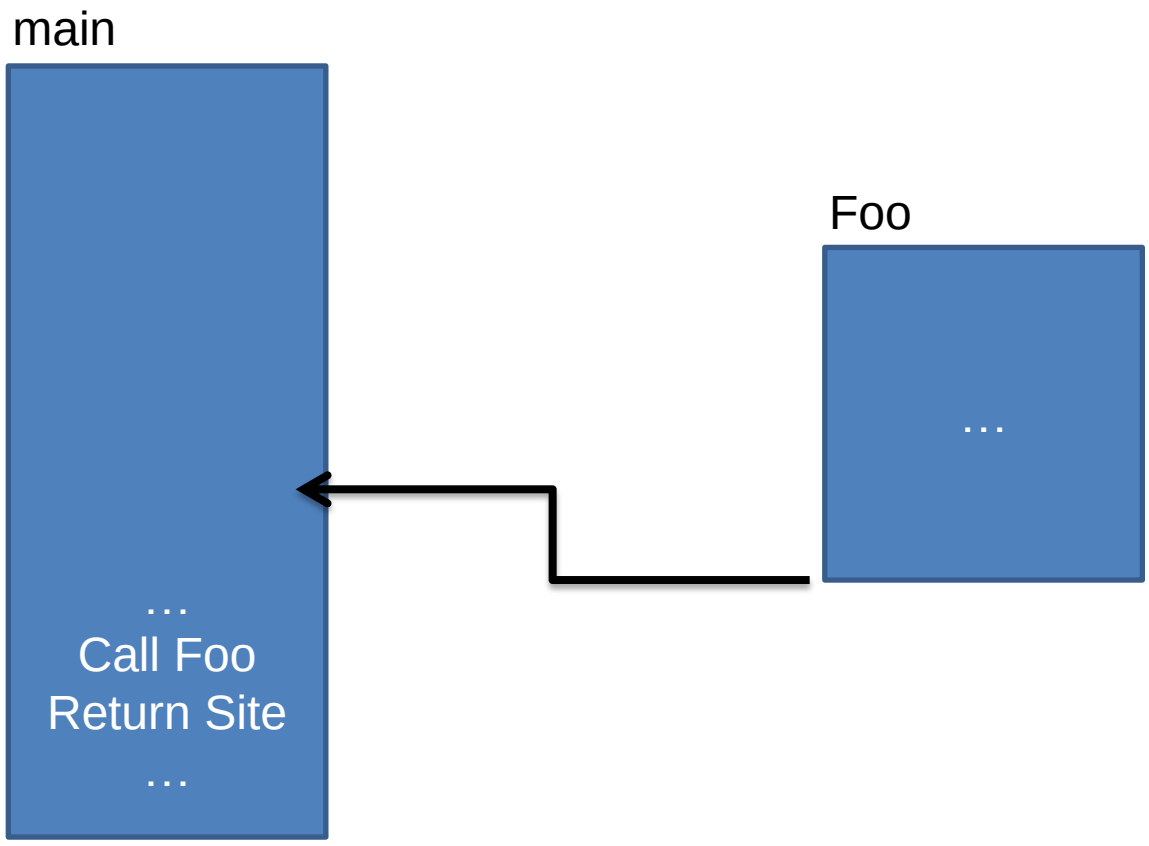
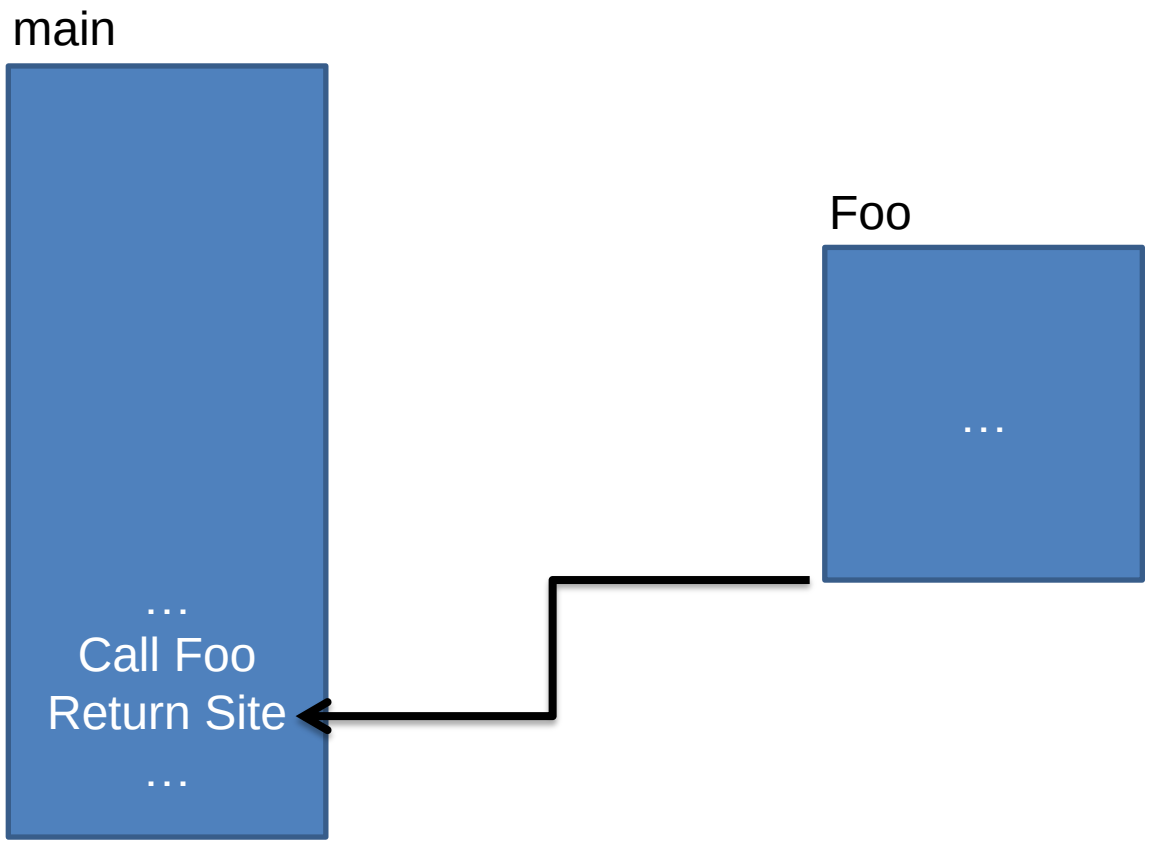


Illustration - Return Address



Optimization

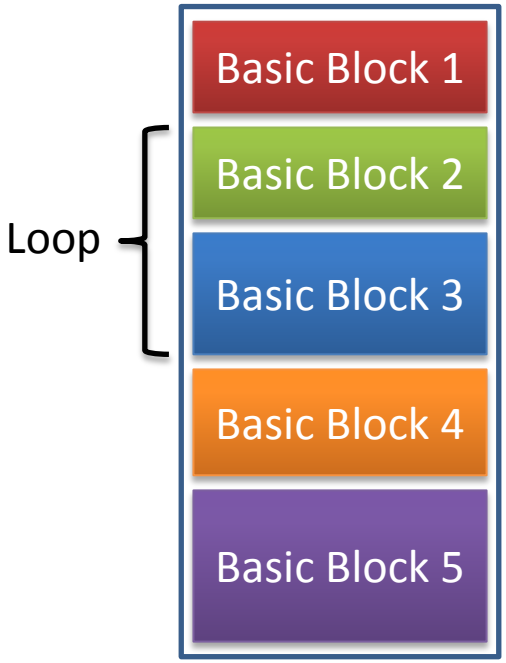
To Speed up the randomization procedure:

- Pre-store the required information
  - Basic block information (e.g., locations)
  - Code/data that need updating

Optimization

To speed up execution:

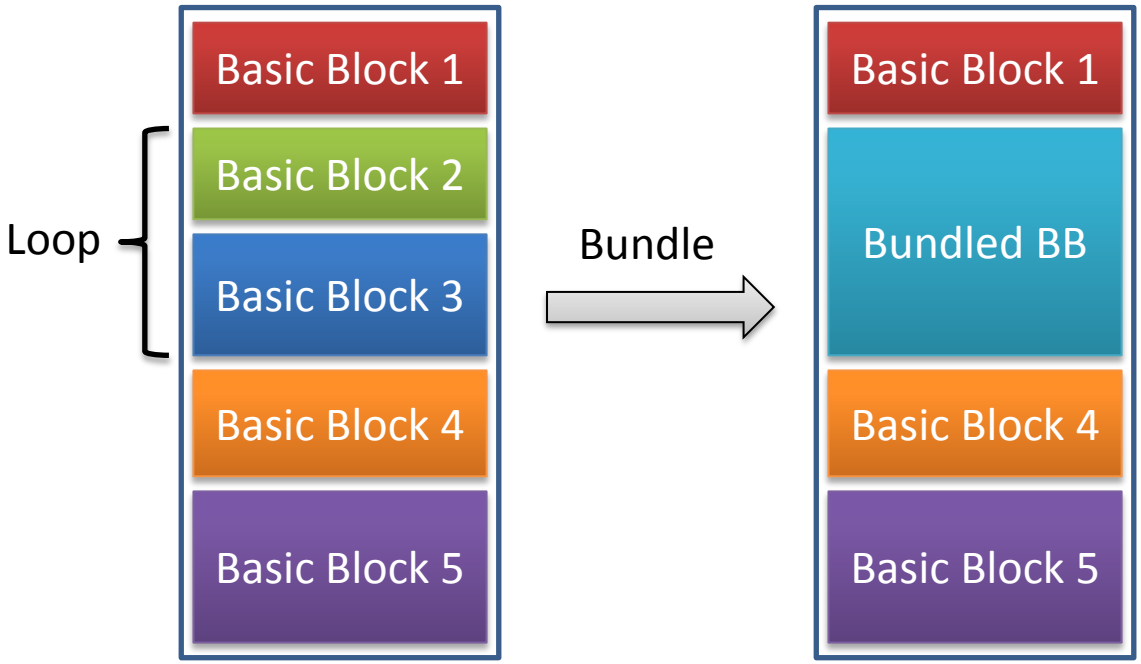
- Probabilistic loop bundling



Optimization

To speed up execution:

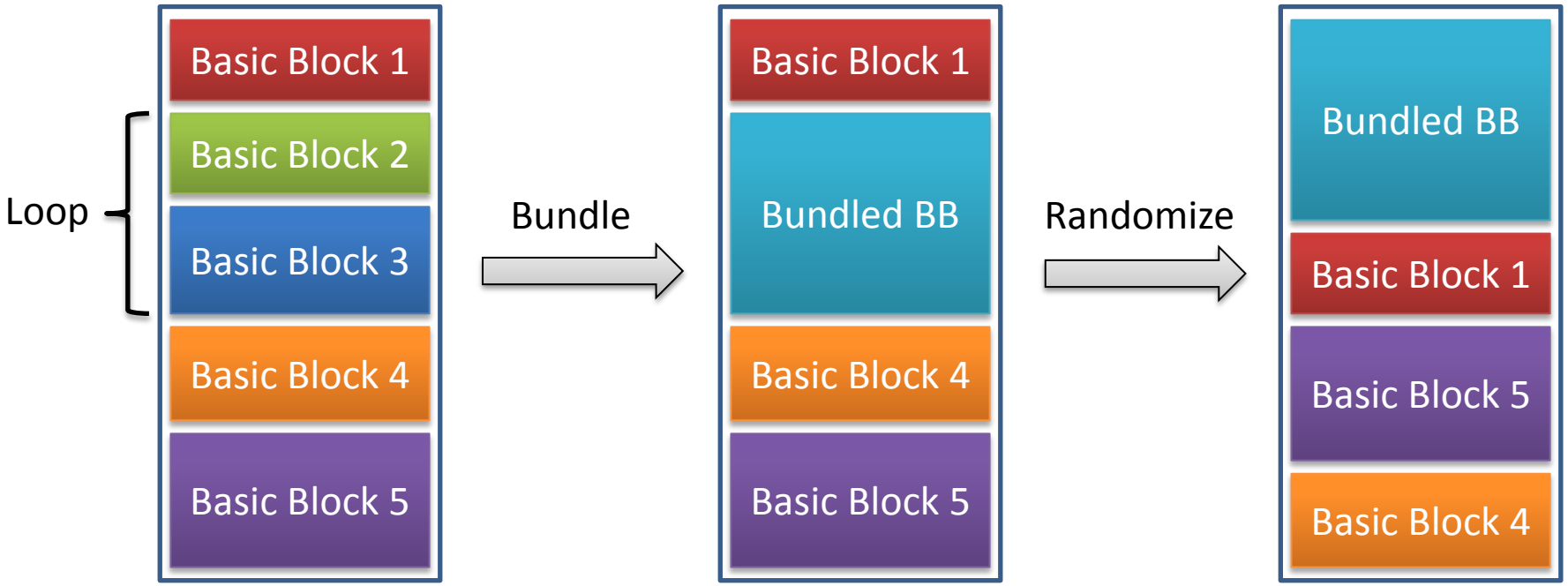
- Probabilistic loop bundling



Optimization

To speed up execution:

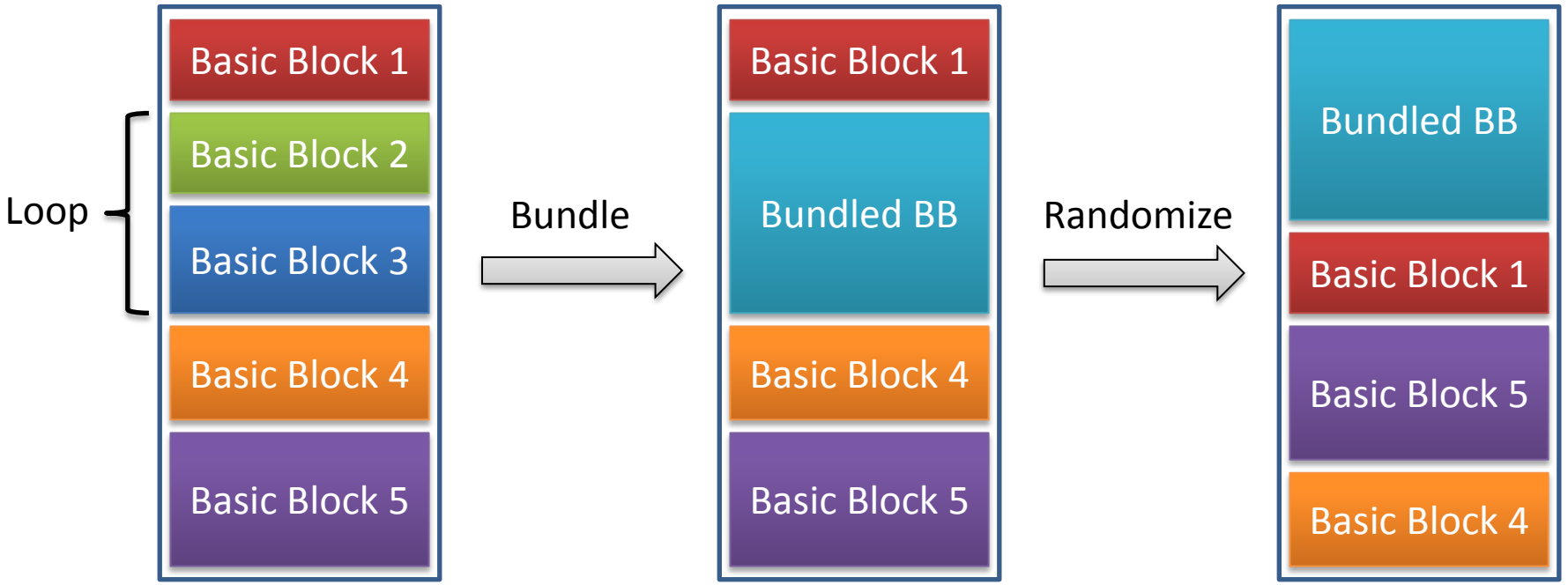
- Probabilistic loop bundling



Optimization

To speed up execution:

- Probabilistic loop bundling



The bundling layout is **different** from time to time. – **Unpredictable**!

Implementation

- Can work on source code using a slightly modified LLVM, or work directly on binaries.
- Can work on Linux user-space applications, and FreeBSD kernel modules.

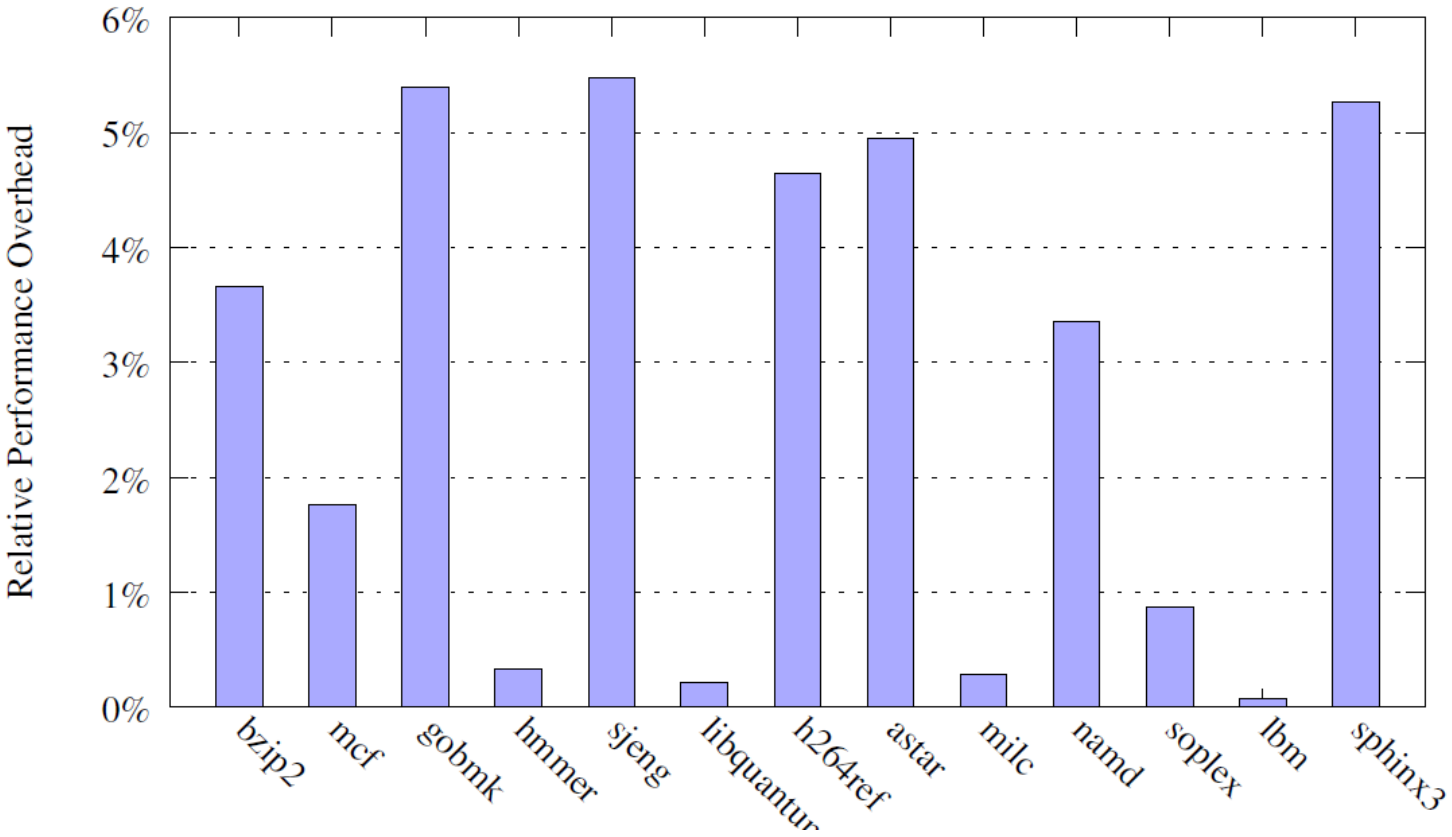
Evaluation - Security

- **Finer-grained randomization:**
  - Adds about four bits of entropy to each instruction.
- **Live randomization during runtime:**
  - Destroy discovered gadgets immediately after each re-randomization.

Software	Apache	nginx	lighttpd
Average Basic Block Number per Function	15.3	18.8	14.4
Average NOP Space (bytes) per Function	19.3	26.2	22.1

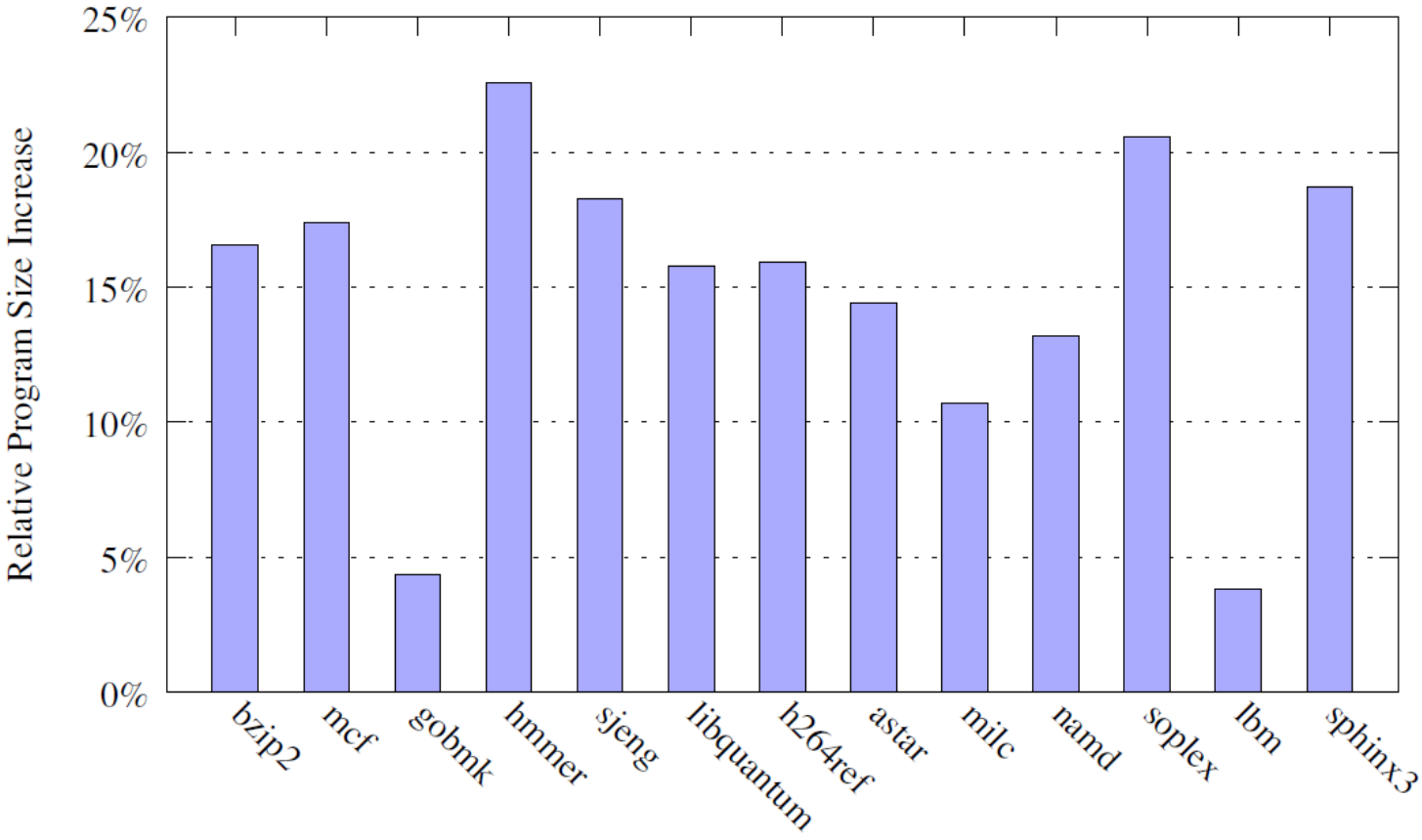
Want more entropy? – Insert more NOP space!

Evaluation - Performance



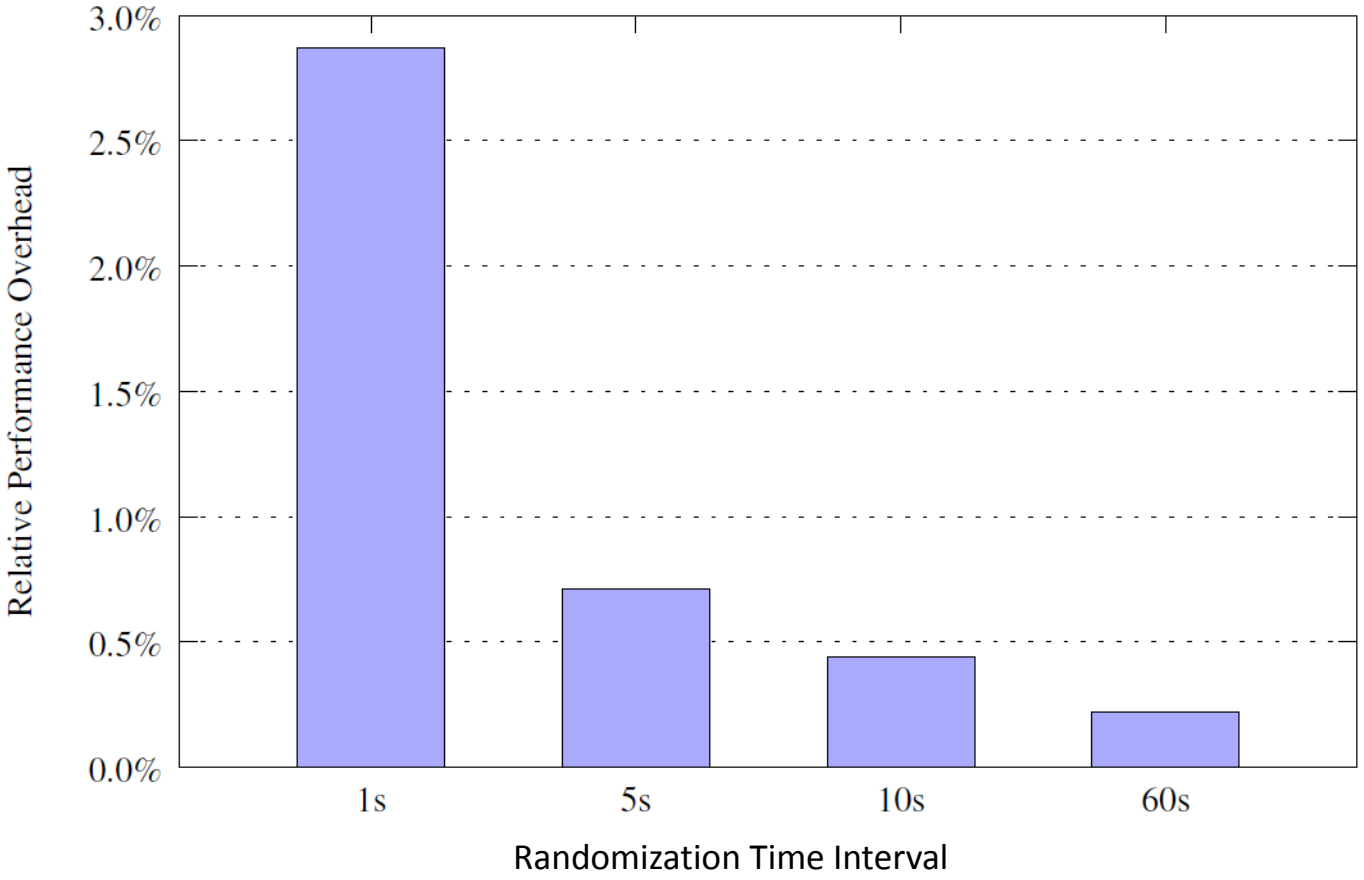
SPEC CPU 2006 Performance Overhead (randomized once)

Evaluation - Performance



SPEC CPU 2006 Size Increase

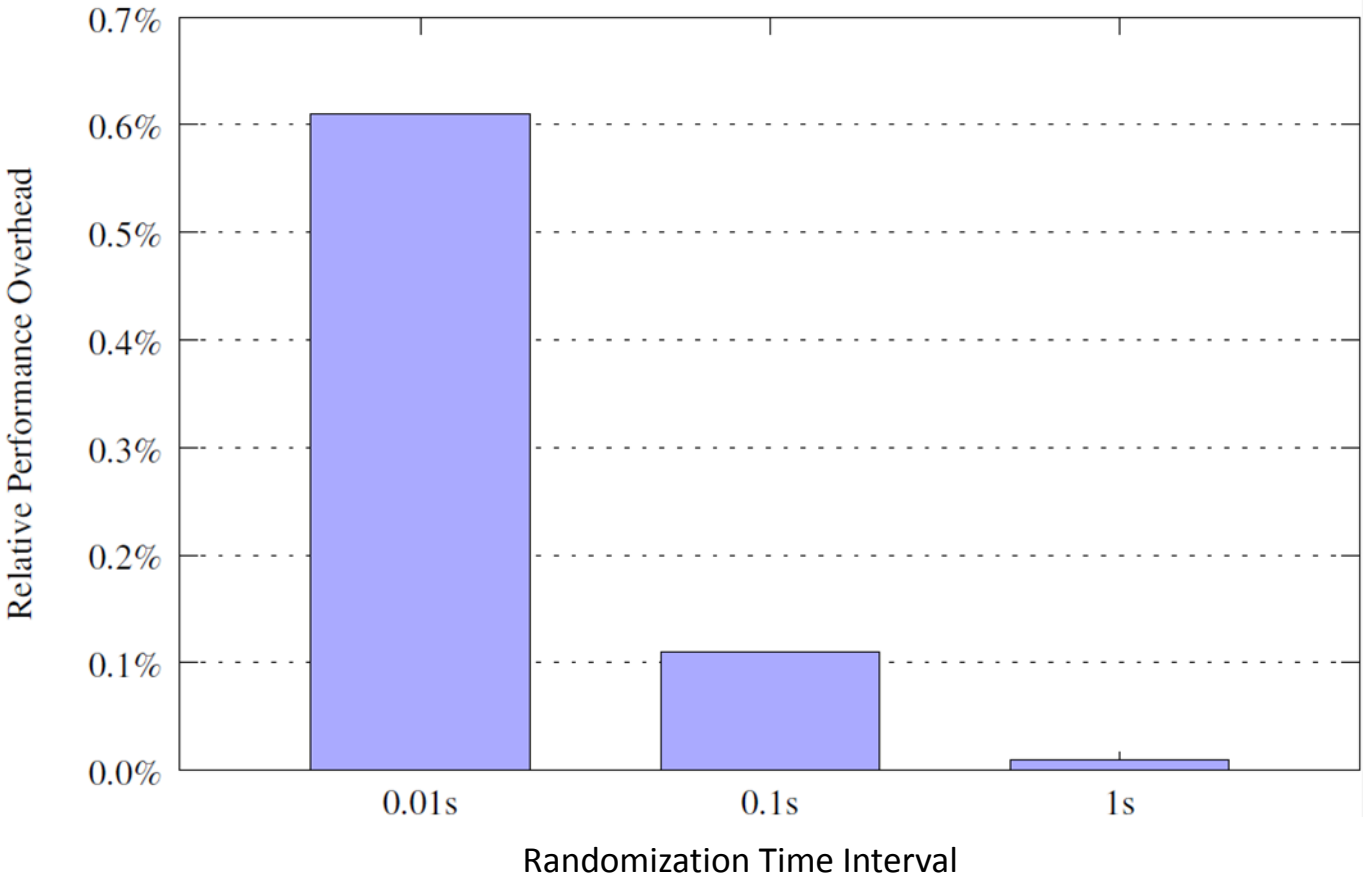
Evaluation - Performance



Apache Web Server Performance Overhead (by ApacheBench)

Performance depends on hardware speed. **Randomization time interval can be random !**

Evaluation - Performance



ReiserFS Performance Overhead (by IOZone)

Performance depends on hardware speed. **Randomization time interval can be random !**

Q&A