Binary Code Continent: Finer-Grained Control Flow Integrity for Stripped Binaries

Minghua Wang[1,2,4], Heng Yin[2], Abhishek Vasisht Bhaskar[2], Purui Su[1,3], Dengguo Feng[1]

1. Chinese Academy of Sciences

2. Syracuse University

3. State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences

4. University of Chinese Academy of Sciences

Background

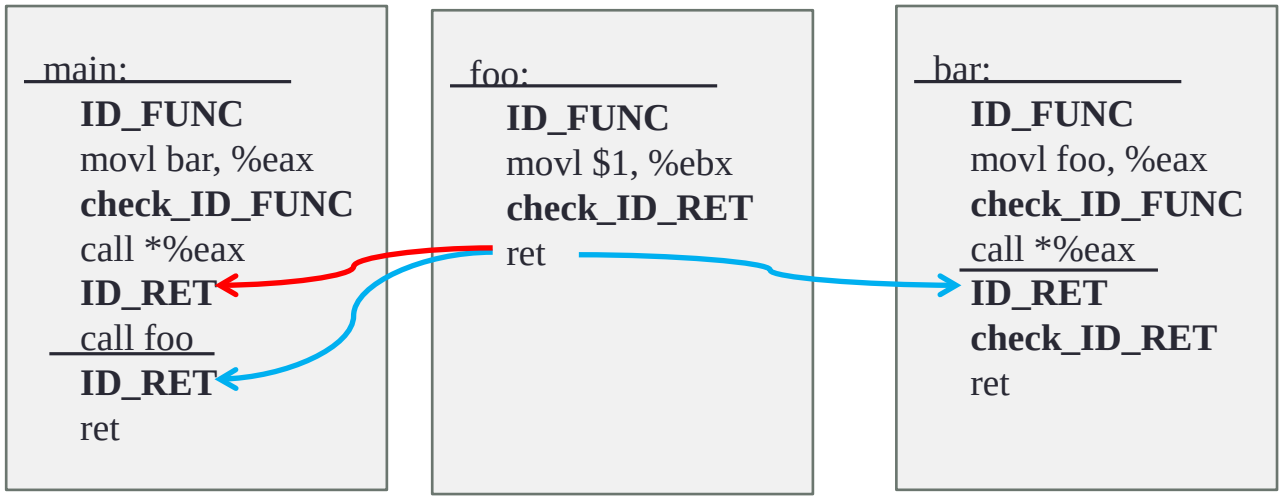
- **Control Flow Integrity**
  - It plays an important role in mitigating control flow hijacking attacks.
  - It forces control flow transfers in the program to follow the policy represented by the CFG.
  - Source code based solutions
    - Fine-grained CFI policy
  - Binary-only solutions
    - Coarse-grained policy
      - Advanced ROP attacks can be launched by leveraging special gadgets

Propose a finer-grained control flow integrity for binaries

Challenges

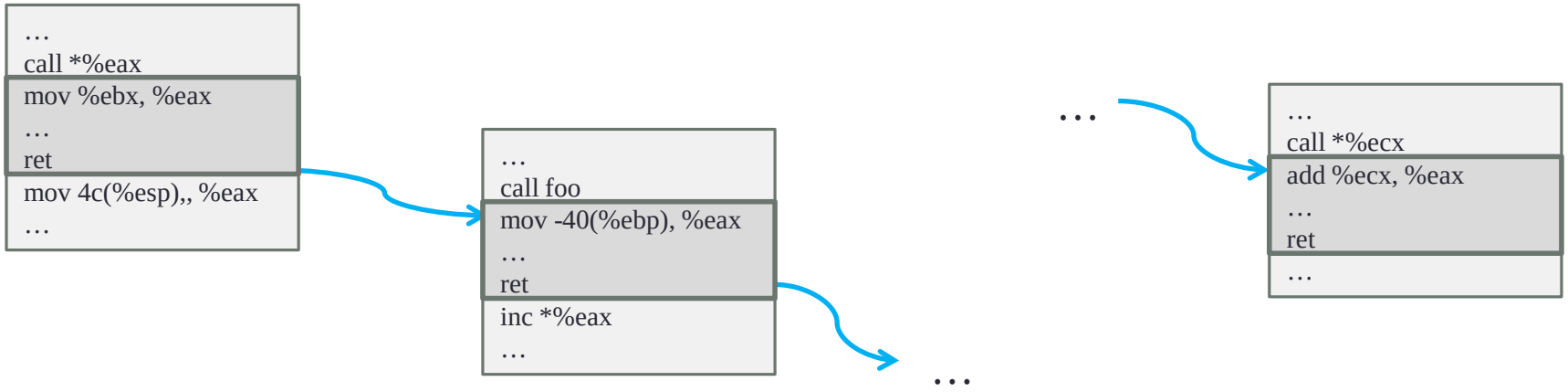
Binary-Only Coarse-Gained CFI

- A return is allowed to reach any callsite

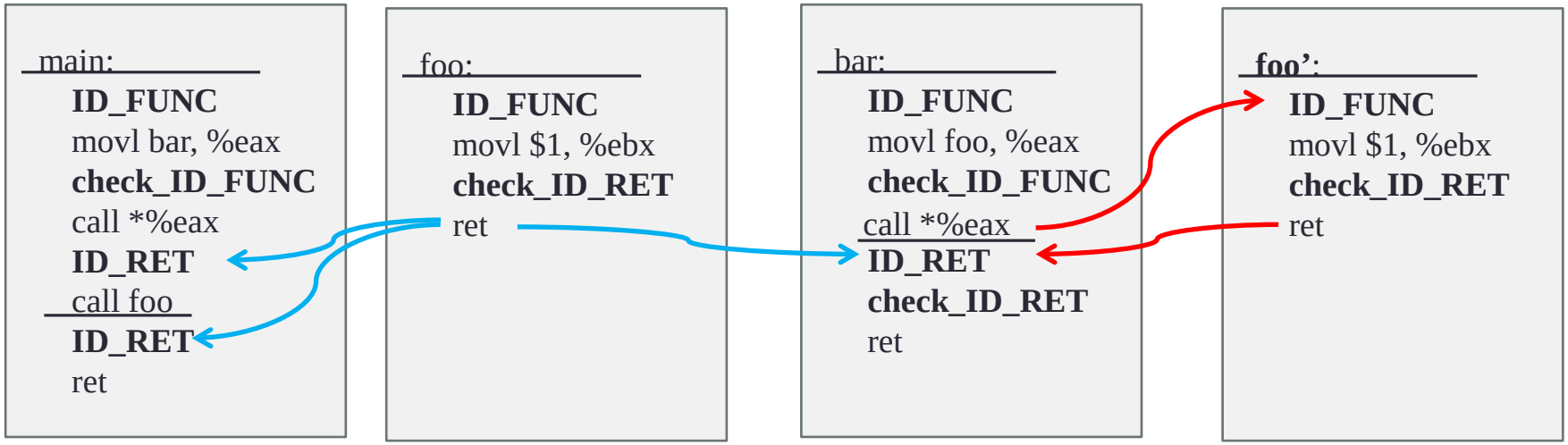


Challenges

- **Bypass BinCFI/CCFIR**
  - Utilize call-preceded gadgets to achieve ROP attacks
    - ✓ Goktas E, et al. ***Out of control: Overcoming control-flow integrity***. S&P 2014.
    - ✓ Carlini N, et al. ***ROP is still dangerous: breaking modern defenses***. USENIX Security,2014.



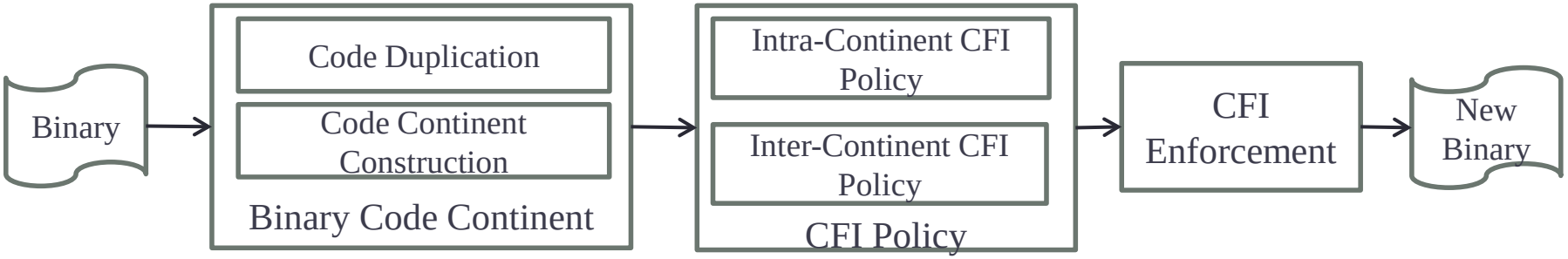
Challenges



- Coarse-Grained CFI：A function is allowed to return to any callsite
- After code duplication and instrumentation：
  - A directly called function will only reach direct callsites
  - An indirectly called function will only reach indirect callsites

Our Approach

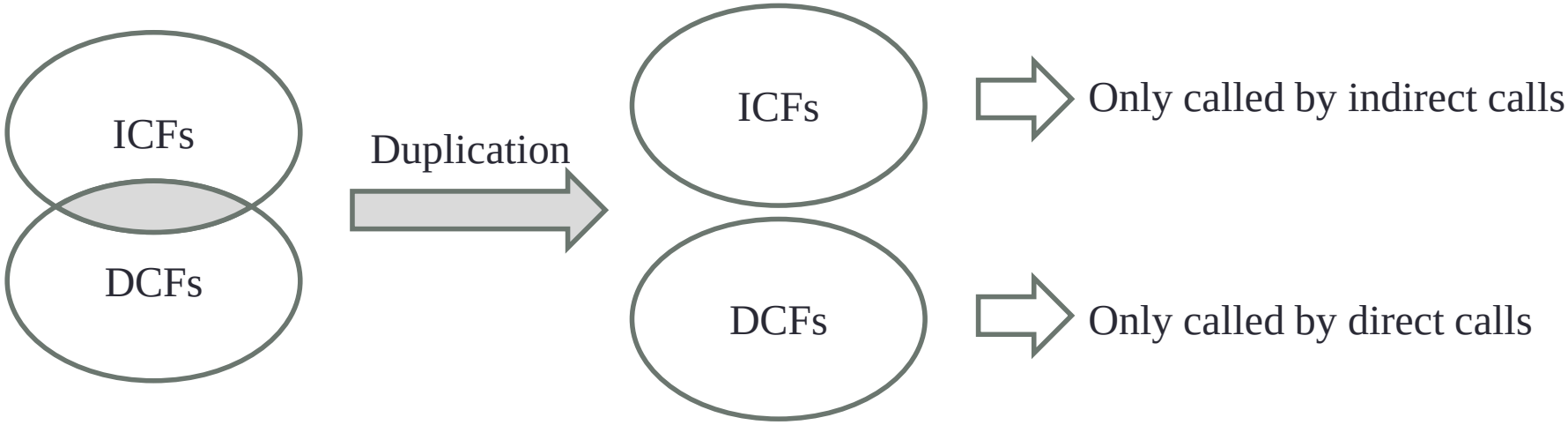
- Duplicate code
- Construct mutually exclusive code continents, which imply the control flow dependency in the binary
- Classify indirect transfers into Intra-Continent and Inter-Continent transfers, and propose Intra-Continent and Inter-Continent CFI policies
- Enforce CFI policies through binary rewriting



Binary Code Continent

Code Duplication

- ICF: Indirect Called Function
  - Constants which imply entry points of functions
- DCF: Direct Called Function
  - Operands of direct callsites



Binary Code Continent

Code Duplication - Example

main:	foo:	bar:	qux:
1: movl bar, %eax	5: movl qux, %eax	8: movl foo, %eax	13: movl $0, %eax
2: call *%eax	6: call *%eax	9: call *%eax	14: ret
3: call foo	7: ret	10: movl $1, %eax	
4: ret		11: call foo	
		12: ret	

Original Functions

foo':
5': movl qux, %eax
6': call *%eax
7': ret

Duplicated Functions

ICF = { foo, qux, bar}

DCF = { foo}

Binary Code Continent

Code Continent

- Describe the control flow dependency in the binary
- Code Continent Construction
  - Construct from SuperCFGs of functions
  - Merge SuperCFGs based on common edges
  - Generate mutually exclusive continents

Binary Code Continent

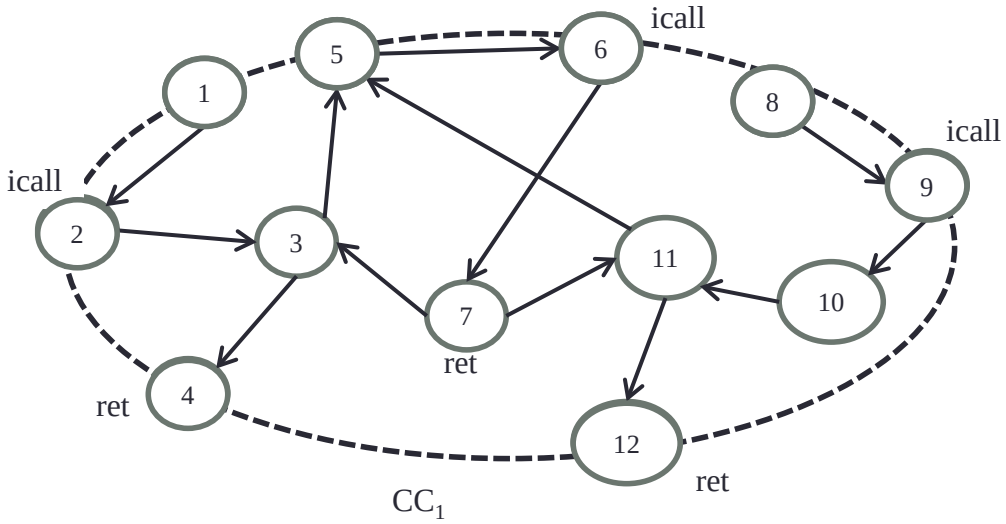
Code Continent – Example

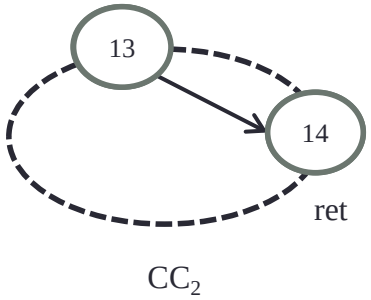
main:	foo:	bar:	qux:
1: movl bar, %eax	5: movl qux, %eax	8: movl foo, %eax	13: movl $0, %eax
2: call *%eax	6: call *%eax	9: call *%eax	14: ret
3: call foo	7: ret	10: movl $1, %eax	
4: ret		11: call foo	
		12: ret	

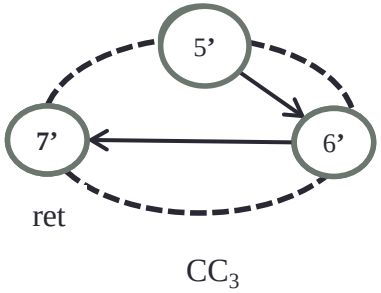
Original Functions

foo':
5': movl qux, %eax
6': call *%eax
7': ret

Duplicated Functions







CFI Policy

- **Continent Node Classification**
- **CFI Policy**
  - Intra-Continent CFI Policy
  - Inter-Continent CFI Policy

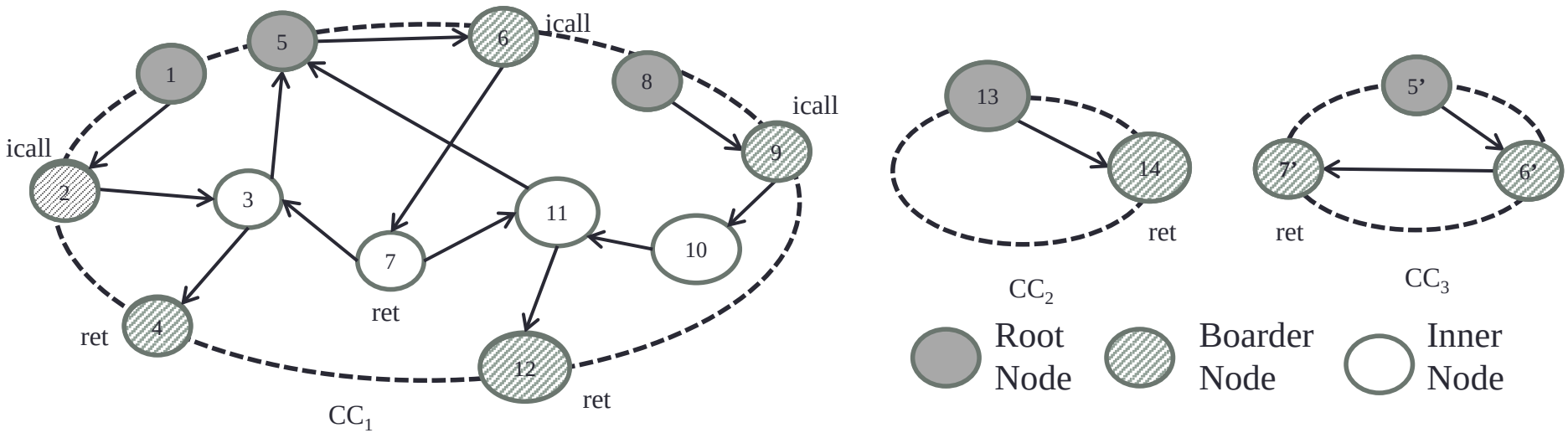
CFI Policy

Continent Node Classification

- Root node：Entry points of ICFs
- Boarder node：Indirect transfers of which targets cannot be statically determined
- Inner node：Nodes except root nodes and boarder nodes

main:	foo:	bar:	qux:
1: movl bar, %eax	5: movl qux, %eax	8: movl foo, %eax	13: movl $0, %eax
2: call *%eax	6: call *%eax	9: call *%eax	14: ret
3: call foo	7: ret	10: movl $1, %eax	
4: ret		11: call foo	
	Original Functions	12: ret	

foo':
5': movl qux, %eax
6': call *%eax
7': ret
Duplicated Functions



CFI Policy

Intra-Continent CFI Policy

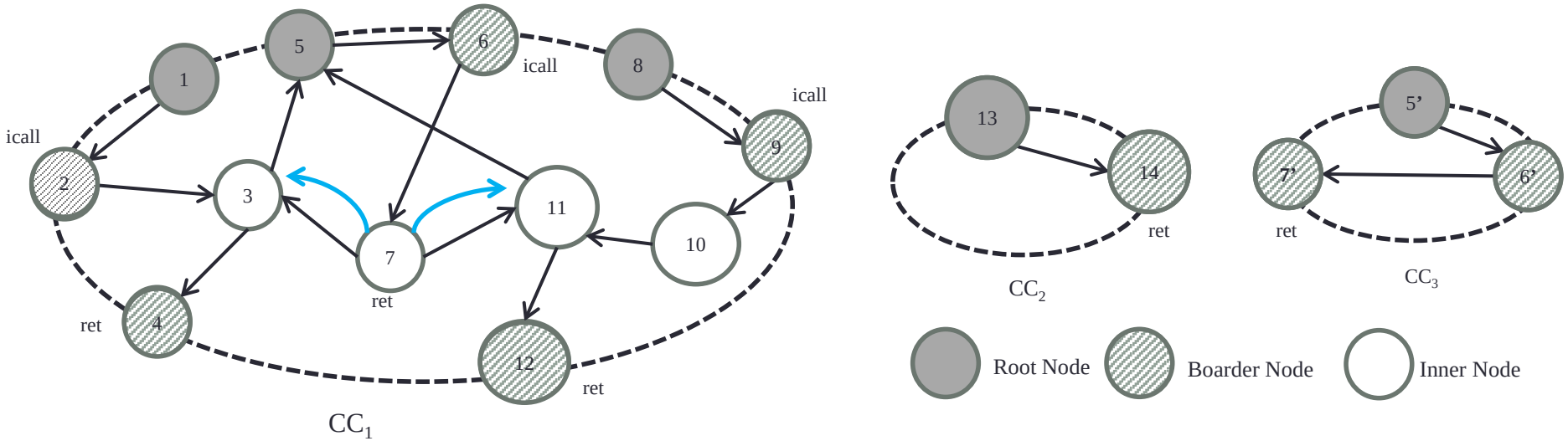
- This policy constrains inner nodes whose targets can be statically determined
- The targets will be always present in current continent, and can be determined by the SuperCFG.

main:	foo:	bar:	qux:
1: movl bar, %eax	5: movl qux, %eax	8: movl foo, %eax	13: movl $0, %eax
2: call *%eax	6: call *%eax	9: call *%eax	14: ret
3: call foo	7: ret	10: movl $1, %eax	
4: ret		11: call foo	
		12: ret	

Original Functions

foo':
5': movl qux, %eax
6': call *%eax
7': ret

Duplicated Functions



CFI Policy

Inter-Continent CFI Policy

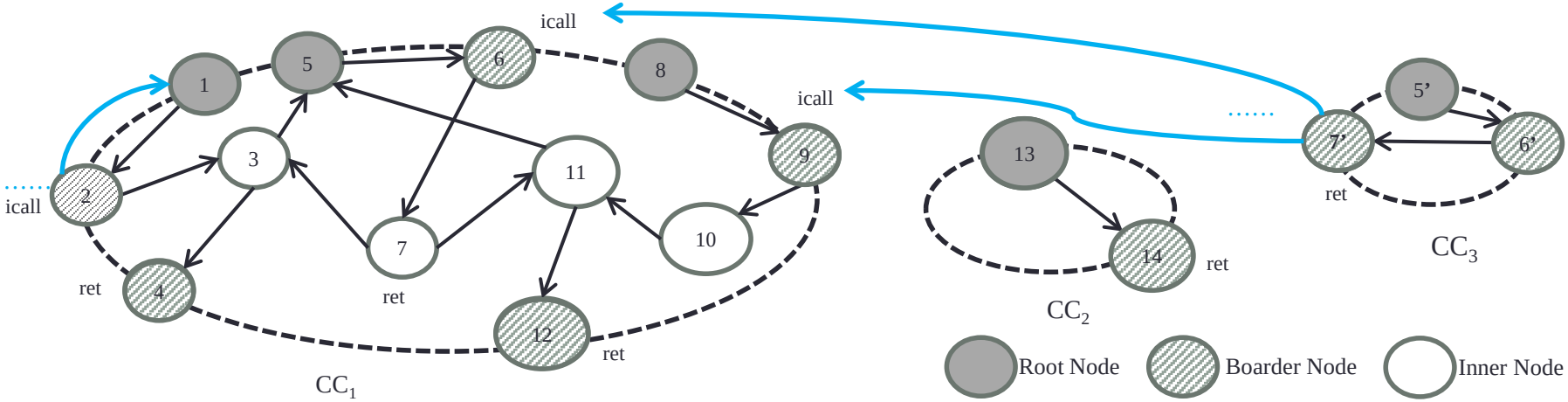
- This policy constrains the boarder nodes:
  - Indirect call nodes can only reach all the root nodes
  - Indirect ret nodes can only reach indirect call nodes
  - Indirect jmp nodes can only reach all the root nodes and call nodes

main:	foo:	bar:	qux:
1: movl bar, %eax	5: movl qux, %eax	8: movl foo, %eax	13: movl $0, %eax
2: call *%eax	6: call *%eax	9: call *%eax	14: ret
3: call foo	7: ret	10: movl $1, %eax	
4: ret		11: call foo	
		12: ret	

Original Functions

foo':
5': movl qux, %eax
6': call *%eax
7': ret

Duplicated Functions



CFI Enforcement

- Instrumentation
- Address Checking Routine
  - Check whether the target is legitimate
  - Each kind of indirect transfers has its own address checking routine
  - In call's routine, the duplicated function will be actually executed if an ICF is called

```
call *%eax      movl %eax, %gs:0x40
                jmp addr_trans_call
```

Instrumentation to indirect calls

```
jmp *%eax       movl %eax, %gs:0x40
                jmp addr_trans_jmp
```

Instrumentation to indirect jmps

```
ret             movl (%esp), %gs:0x40
                jmp addr_trans_ret
```

Instrumentation to indirect returns

```
_addr: ret   _addr:  prefetchnta start
                     prefetchnta size
                     movl _addr, %gs:0x50
                     movl %(esp), %gs:0x40
                     jmp addr_trans_dret
```

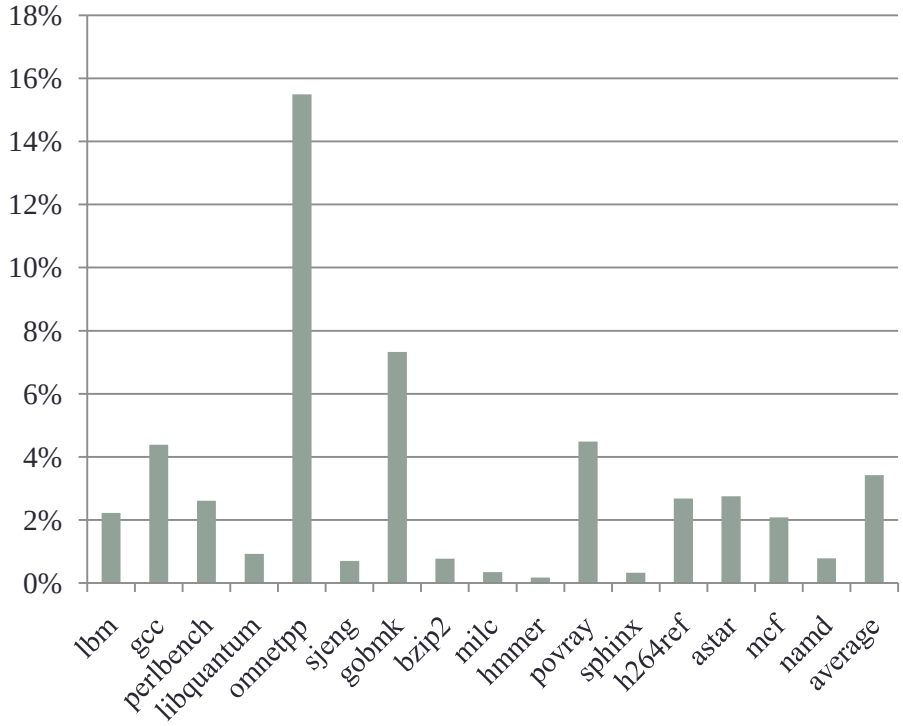
Instrumentation to direct returns

Evaluation

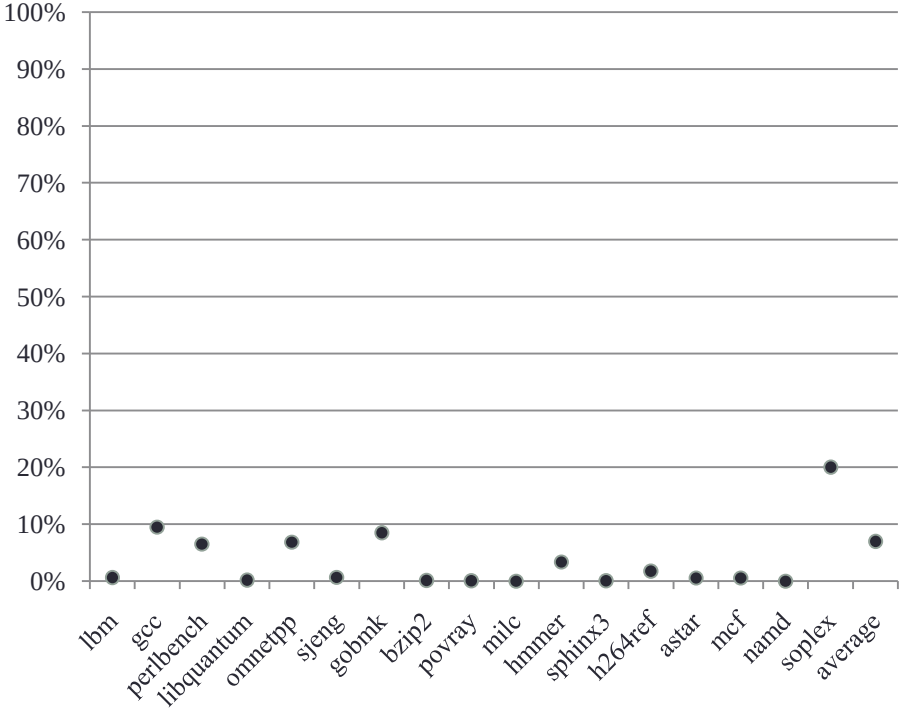
- Implementation
  - BinCC (Binary Code Continent)
- Evaluate BinCC from four respects
  - Code Duplication Evaluation
  - Indirect Transfer Targets Deduction
  - ROP Attack Evaluation
  - Performance

Evaluation

Code Duplication Evaluation



Percentage of Duplicated Functions



Percentage of Duplicated Instructions

Evaluation

Indirect Transfer Targets Reduction

- AIR(Average Indirect target Reduction)，proposed by BinCFI：

$$AIR = \frac{1}{N}\sum_{j=1}^{N}\left(1 - \frac{|T_j|}{S}\right)$$

T_j ：the legitimate targets of indirect transfer i_j
S ：size of code section
N ：the number of all the indirect transfers

BinCFI：98.86%，BinCC: 99.54%

- A new metric: **RAIR (Relative AIR)**, to evaluate extent to which BinCC refines the legitimate targets compared to BinCFI：

$$RAIR = \frac{1}{N}\sum_{j=1}^{N}\left(1 - \frac{|T_j|}{|T_j'|}\right)$$

T_j ：the legitimate targets of i_j in BinCFI
T_j' ：the legitimate targets of i_j in BinCC
N ：the number of all the indirect transfers

BinCC reduced legitimate targets by **81.34%** from BinCFI

Evaluation

ROP Attacks Evaluation

- GS (Gadget Survivability):

  Suppose in a CFI enforced binary a return instruction was fully controlled by attackers, how likely this return could be used to reach a call preceded gadget in this binary

$$GS = \frac{1}{|R|}\sum_{i=0}^{|R|}\frac{|C_i|}{|C|}$$

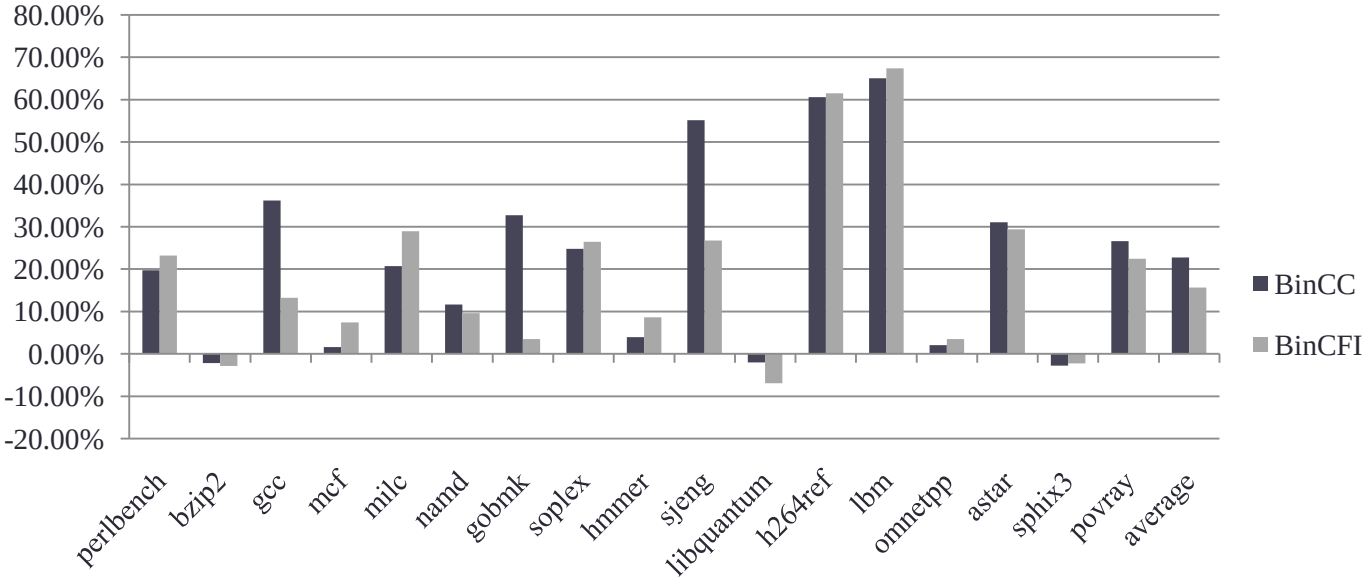
C_i : the legitimate targets of ret_i in a CFI enforced binary
R : all the returns in the binary
C : all the call-preceded gadgets in the binary

BinCFI: 100%, BinCC: 0.70%

Evaluation

Performance

- Space Overhead
  - 125% of the original binary size
  - 14% lower than the size of BinCFI-enforced binaries
- Runtime Overhead



Conclusion

- Propose binary code continent as a way to enforce finer-grained CFI policy in x86 stripped ELF binaries
- Classify indirect transfers into Intra-Continent and Inter-Continent transfers, and apply separate, strict policies to constrain them
- Significantly reduce the legitimate transfer targets by 81.34% from BinCFI
- Significantly increase the difficulty for adversaries to bypass CFI to launch sophisticated ROP attacks

Thanks !!

Q & A