Fool Me If You Can:

On the Robustness of Binary Code Similarity Detection Models against Semantics-preserving Transformations

Jiyong Uhm • **Minseok Kim** • **Michalis Polychronakis** • **Hyungjoon Koo**

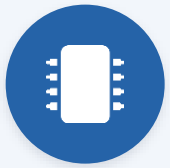
Sungkyunkwan University; Stony Brook University

Understanding Software

Binaries are everywhere


Servers


Mobile


System


Cloud

Binary analysis can…


Forensics


Malware


Vulnerability


Reverse Engineering

Hard to scale


Expert

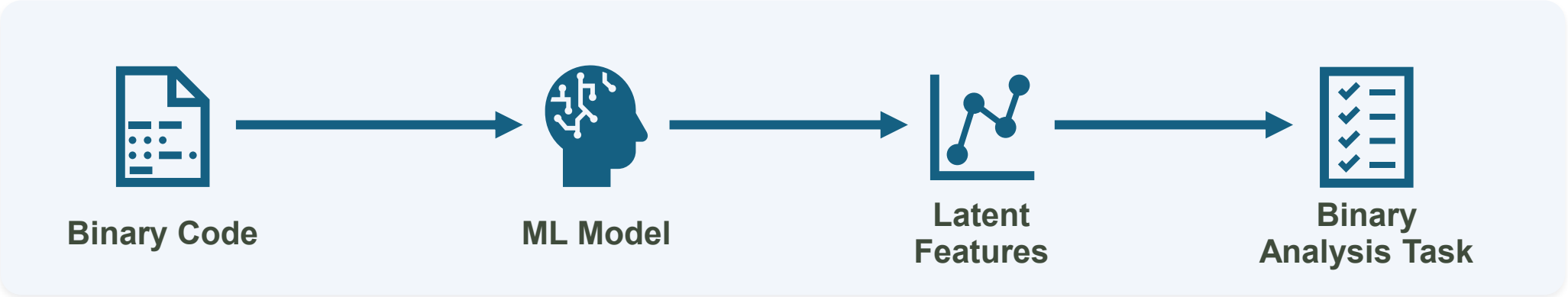

Slow


Anti Reversing

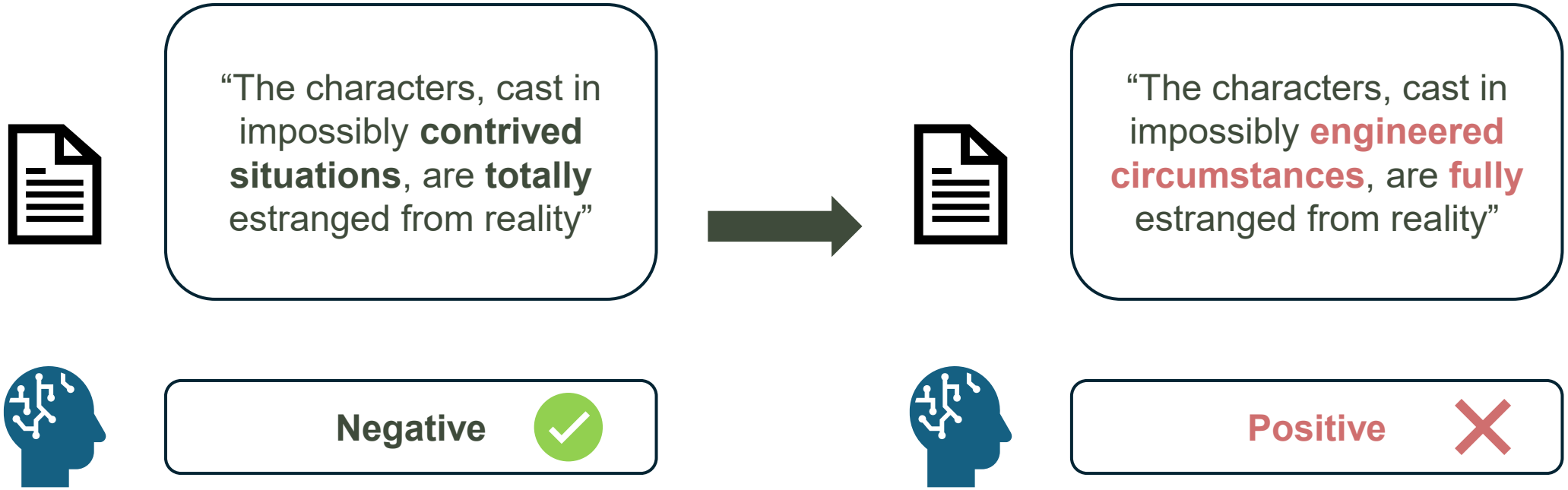

Manual

Applying ML to binary analysis

Binary analysis via machine learning

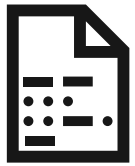


Model Robustness in NLP (*TextFooler*, AAAI '20)

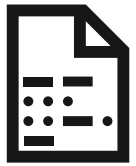


Jin, Di, et al. "Is bert really robust? a strong baseline for natural language attack on text classification and entailment." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. No. 05. 2020.

Models for Assembly?



```
mov     eax, edi
add     eax, 5
mov     ecx, eax
sub     ecx, esi
test    ecx, ecx
```



```
lea     eax, [edi + 5]
mov     ecx, eax
sub     ecx, esi
test    ecx, ecx
```

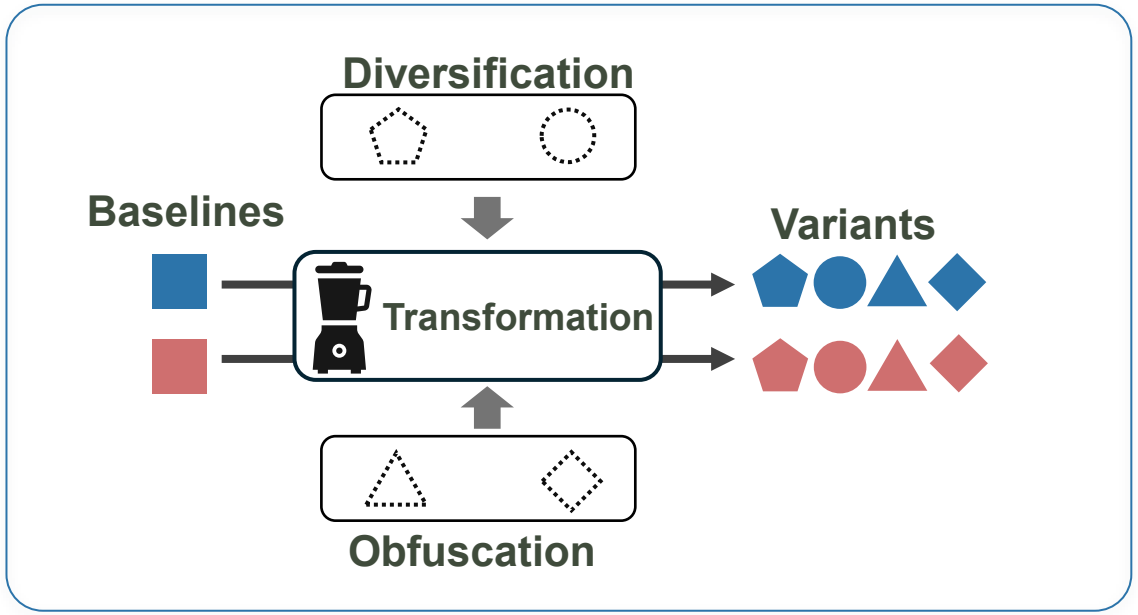
Negative 

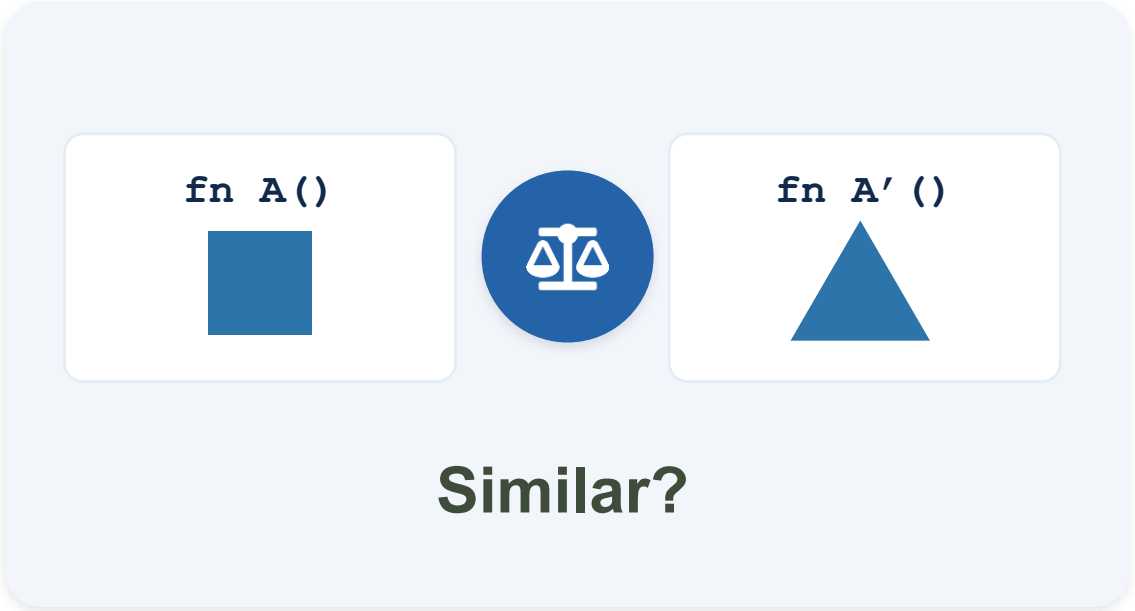
Positive 

Task: Binary Code Similarity Detection

Binary Code Similarity Detection (BCSD)

- Infer whether a piece of code originates from the same source
- Multiple existing work + Numerous Applications





: fn A() : fn B()

Target ML Models

Graph Neural Network

Gemini [1]
Genius [2]

Context independent embedding

SAFE [3]
Asm2Vec [4]

Context dependent embedding (BERT-based)

Trex [5]
BinShot [6]

[1] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. 2017. Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)
[2] Qian Feng, Rundong Zhou, Chengcheng Xu, Yao Cheng, Brian Testa, and Heng Yin. 2016. Scalable Graph-based Bug Search for Firmware Images. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16).
[3] Luca Massarelli, Giuseppe Antonio Di Luna, Fabio Petroni, Roberto Baldoni, and Leonardo Querzoni. 2019. SAFE: Self-Attentive Function Embeddings for Binary Similarity. In Proceedings of Detection of Intrusions and Malware, and Vulnerability Assessment: 16th International Conference (DIMVA '19).
[4] Steven H. H. Ding, Benjamin C. M. Fung, and Philippe Charland. 2019. Asm2Vec: Boosting Static Representation Robustness for Binary Clone Search against Code Obfuscation and Compiler Optimization. In 2019 IEEE Symposium on Security and Privacy (S&P '19).
[5] Kexin Pei, Zhou Xuan, Junfeng Yang, Suman Jana, and Baishakhi Ray. 2022. Trex: Learning execution semantics from micro-traces for binary similarity. IEEE Transactions on Software Engineering (2022).
[6] Sunwoo Ahn, Seonggwan Ahn, Hyungjoon Koo, and Yunheung Paek. 2022. Practical Binary Code Similarity Detection with BERT-based Transferable Similarity Learning. In Proceedings of the 38th Annual Computer Security Applications Conference (ACSAC '22).

Previous Approaches

Capozzi et al [7] and FuncFooler [8]

- Implement transformation as source code perturbation (using inline assembly)

```
int add(int x, int y){
    int z = x + y;
    return z * 2;
}
```

```
int add(int x, int y){
    int z = x + y;
    asm volatile (
        "add $5, %0\n\t"
        "sub $5, %0\n\t"
        :"+r"(z)
    );
    return z * 2;
}
```

[7] Gianluca Capozzi, Daniele Cono D'Elia, Giuseppe Antonio Di Luna, and Leonardo Querzoni. 2024. Adversarial attacks against binary similarity systems. IEEE Access (2024).
[8] Lichen Jia, Bowen Tang, Chenggang Wu, Zhe Wang, Zihan Jiang, Yuanming Lai, Yan Kang, Ning Liu, and Jingfeng Zhang. 2022. FuncFooler: A Practical Black-box Attack Against Learning-based Binary Code Similarity Detection Methods. arXiv preprint arXiv:2208.14191 (2022).

asmFooler

Generate semantics-preserving variants ➔ Test robustness of ML Models

Baselines

620 real-world ELF binaries

8 Transformations

Diversification and Obfuscation

Variants

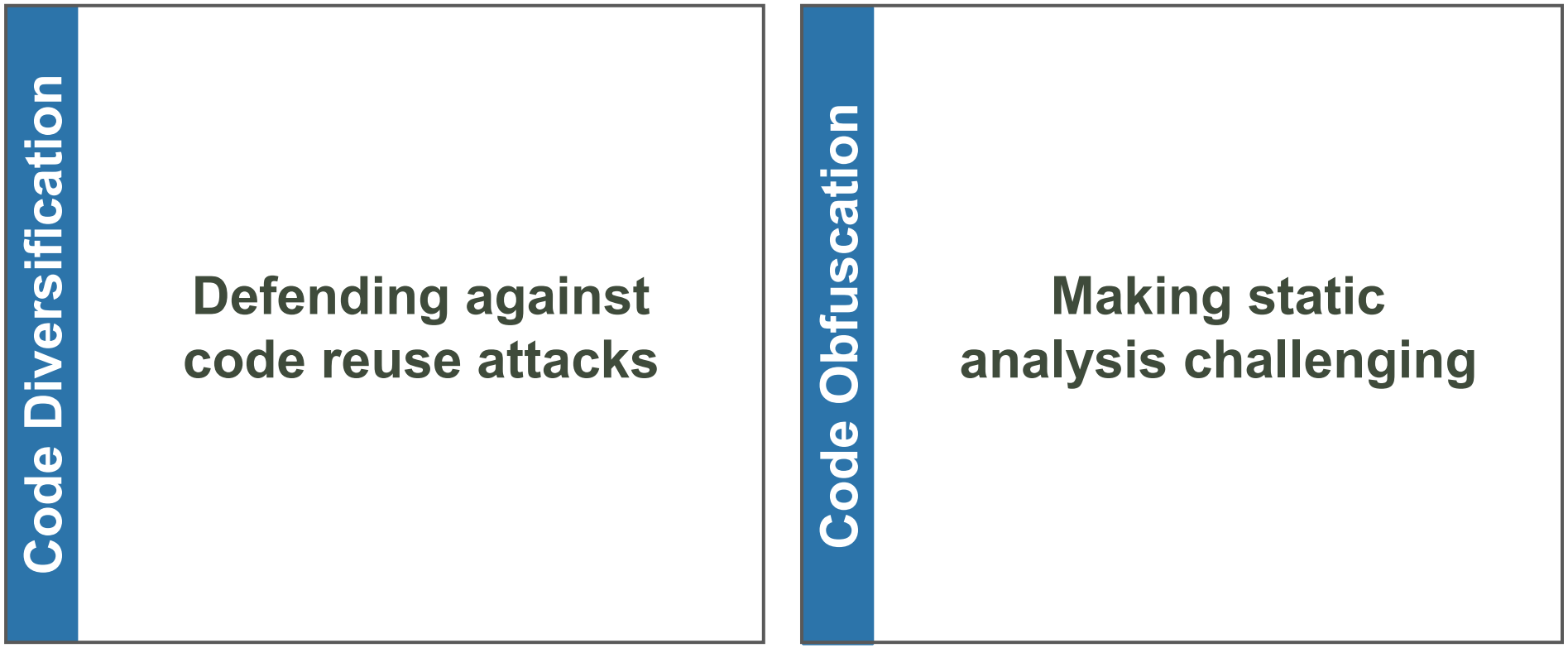
9,565 adversarial binaries

6 BCSD Models

FN / FP triggering perturbation

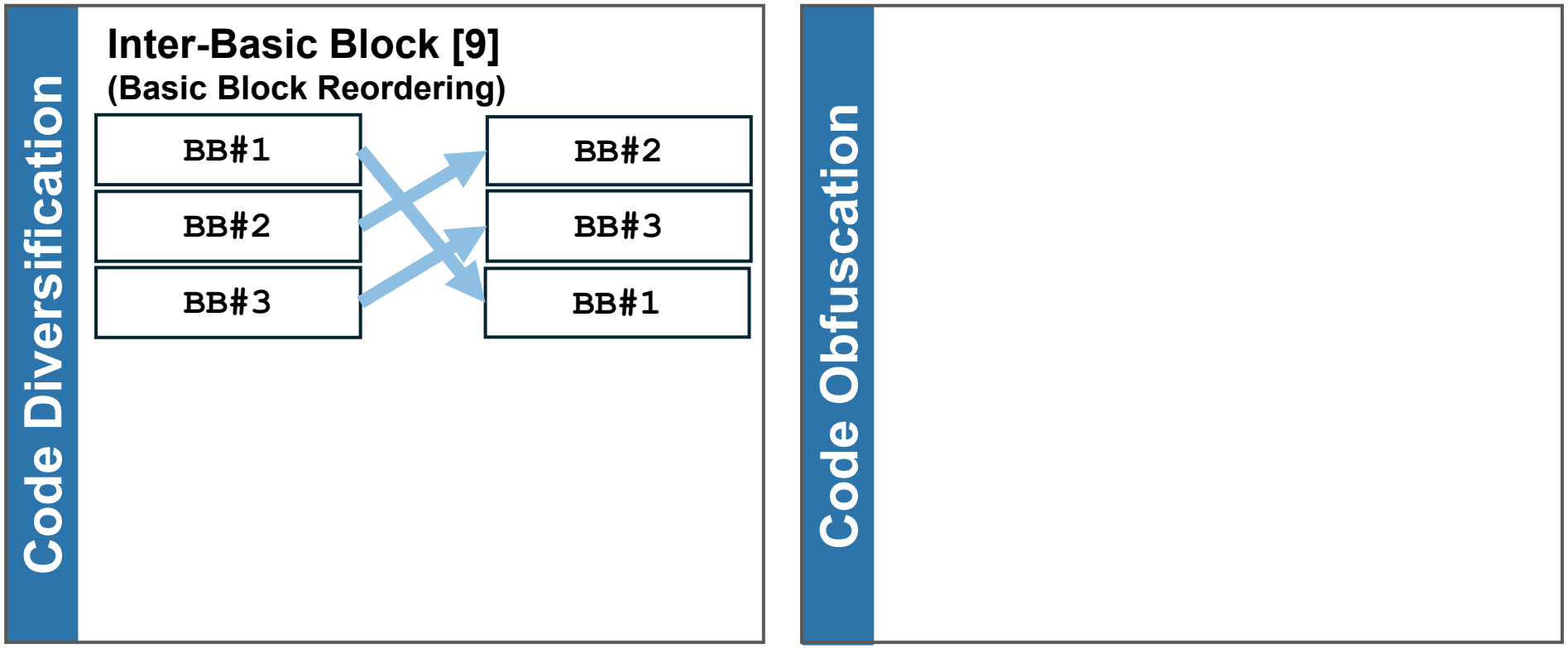
Semantics-Preserving Transformations

- Transformation that preserve semantics



Semantics-Preserving Transformations

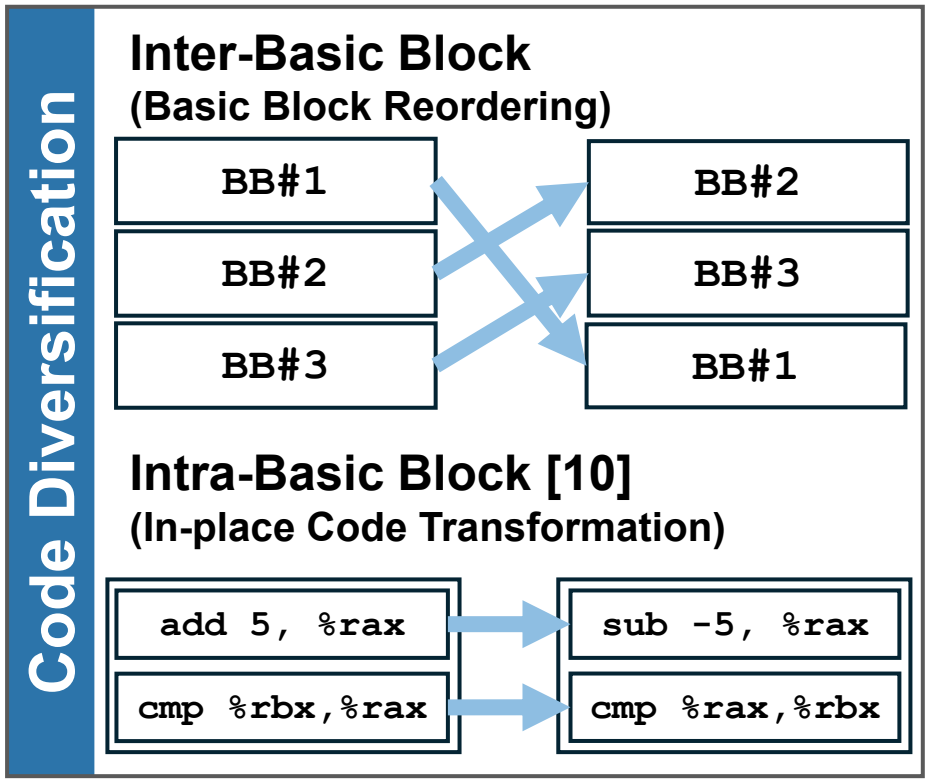
- Reordering between basic blocks

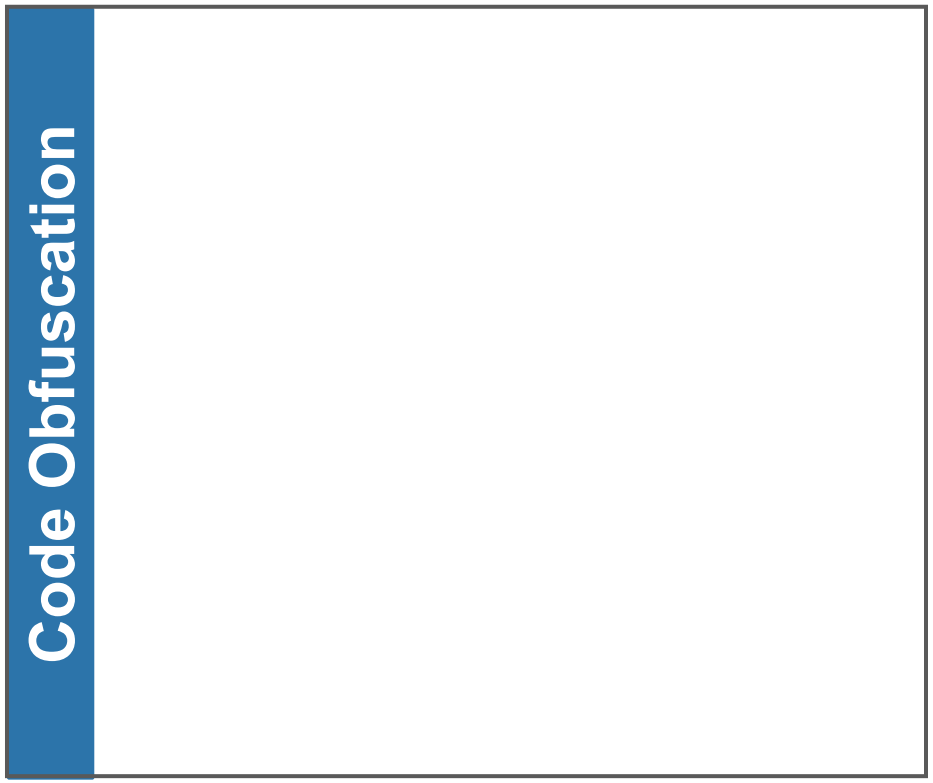


[9] Hyungjoon Koo, Yaohui Chen, Long Lu, Vasileios P. Kemerlis, and Michalis Polychronakis. 2018. Compiler-Assisted Code Randomization. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (S&P '18).

Semantics-Preserving Transformations

- Reordering within basic blocks

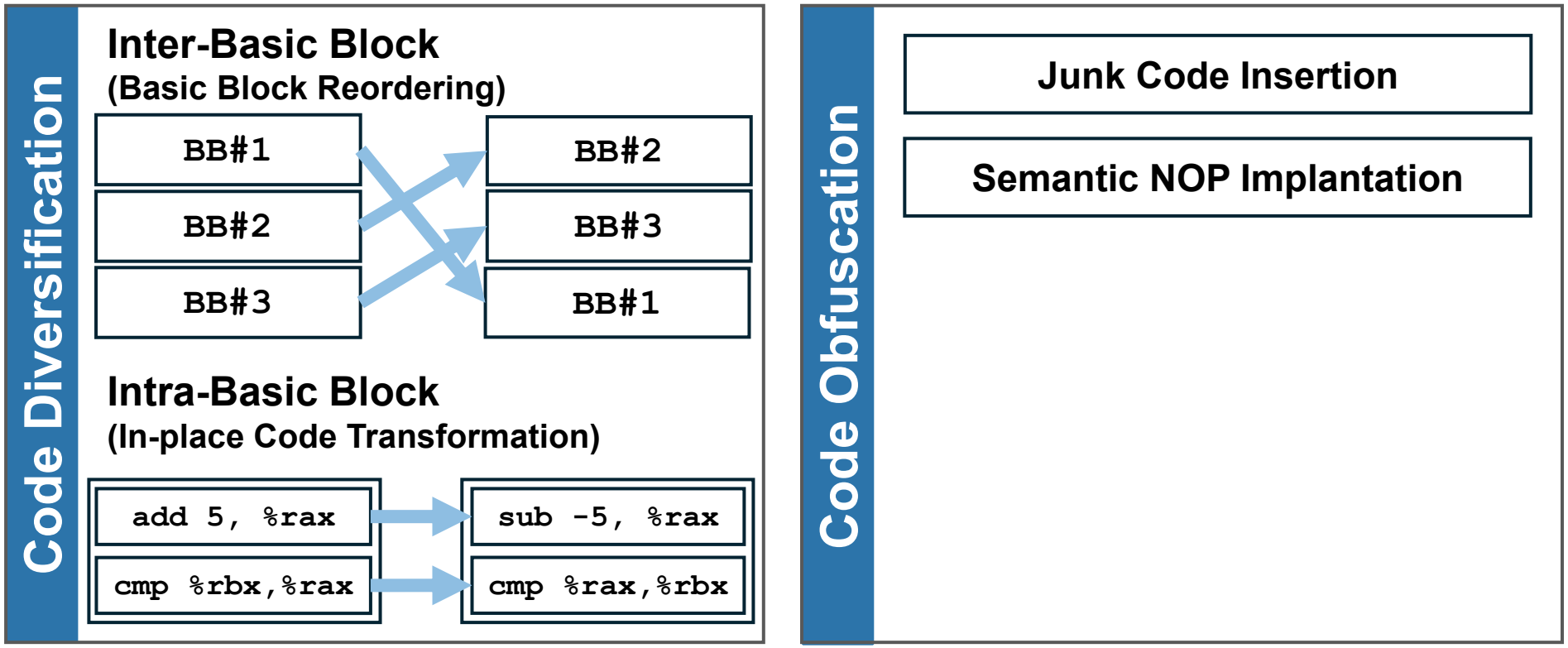




[10] Vasilis Pappas, Michalis Polychronakis, and Angelos D. Keromytis. 2012. Smashing the Gadgets: Hindering Return Oriented Programming Using In-place Code Randomization. In Proceedings of the 2012 IEEE Symposium on Security and Privacy (S&P '12).

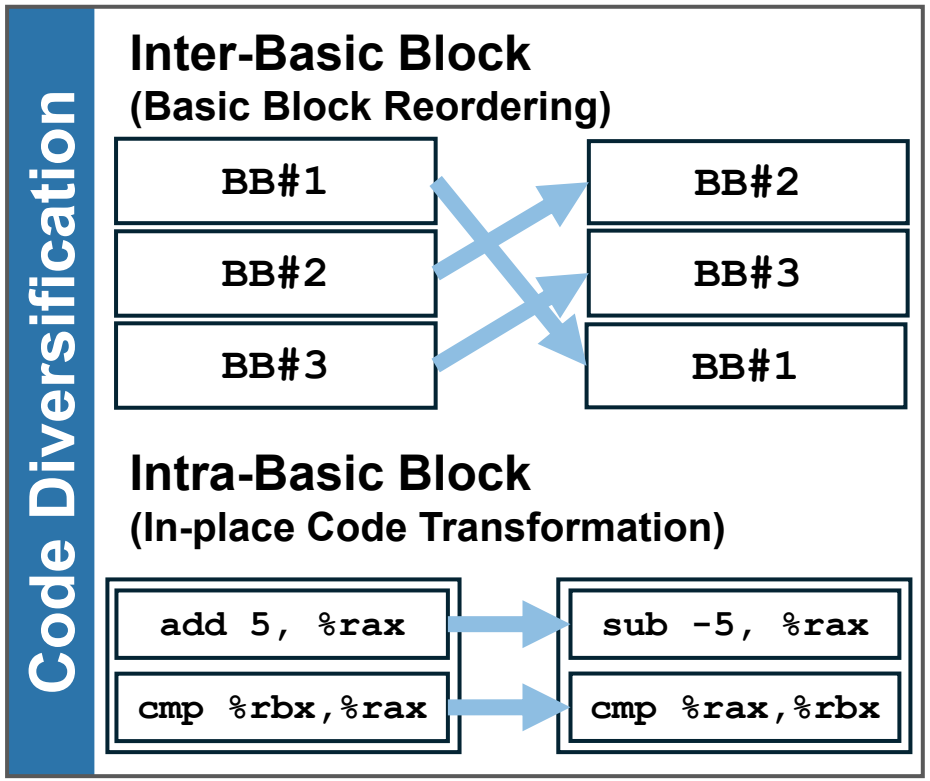
Semantics-Preserving Transformations

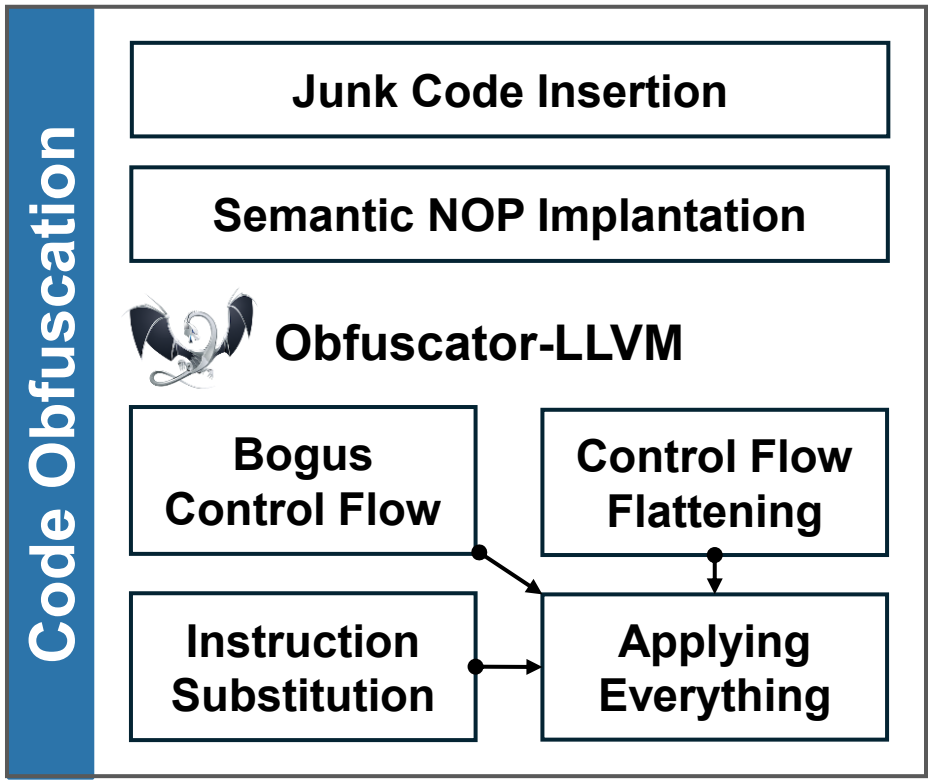
- Addition of inconsequential sequence of instructions



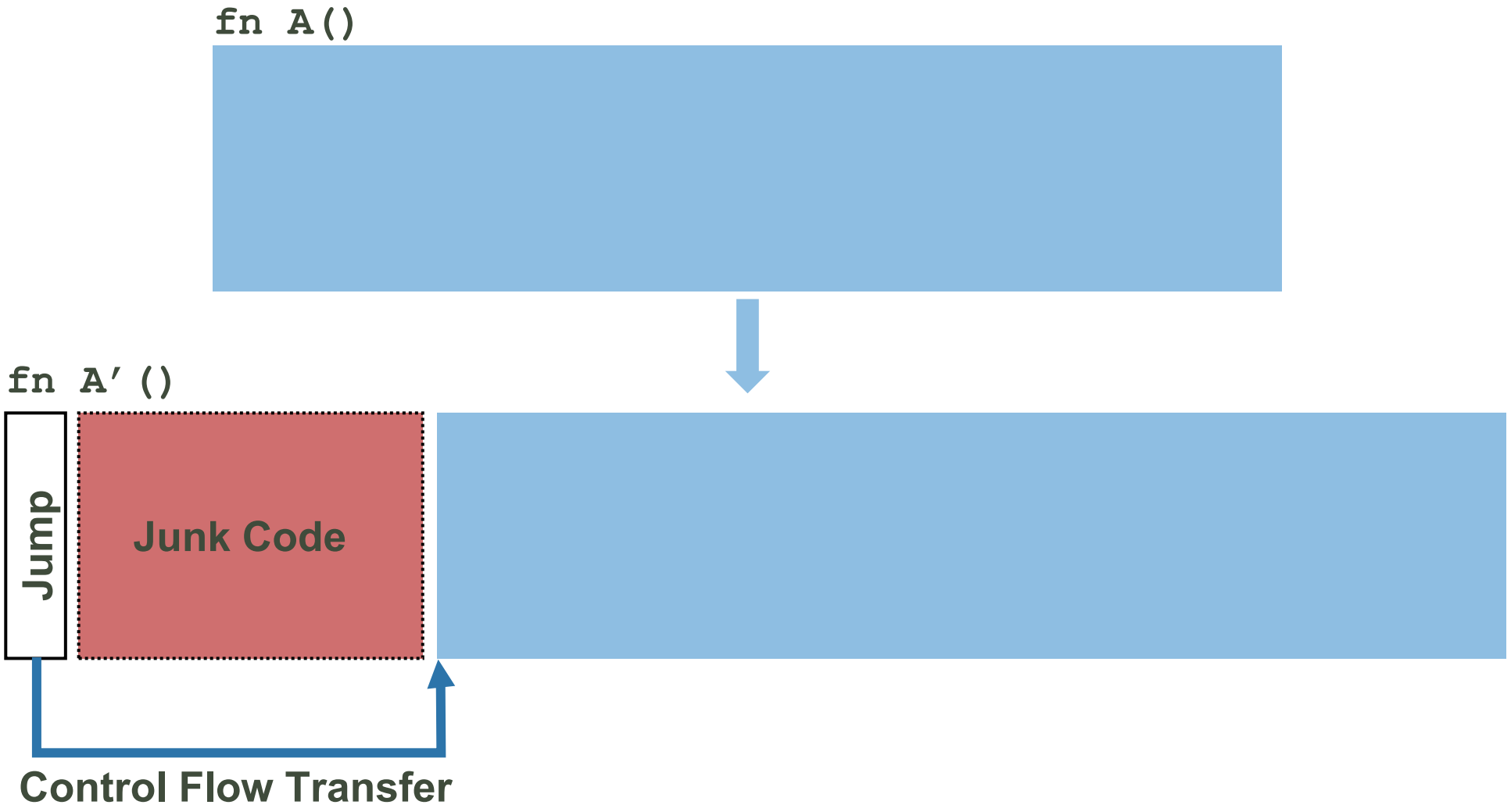
Semantics-Preserving Transformations

- Obfuscator LLVM





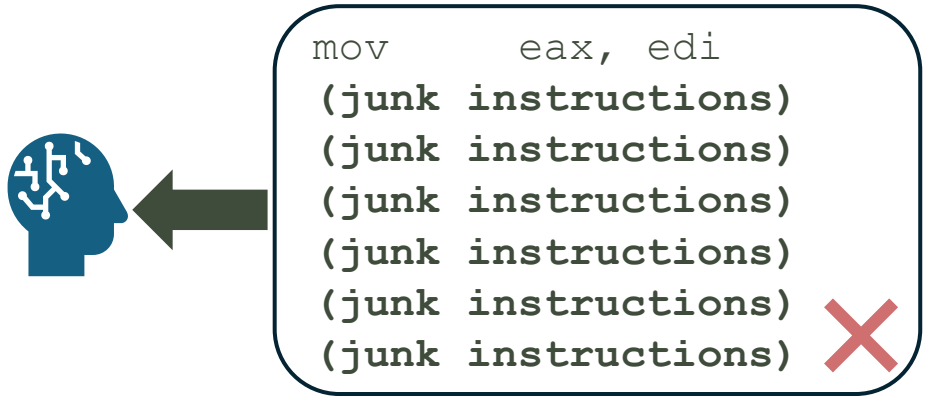
Junk Code Insertion

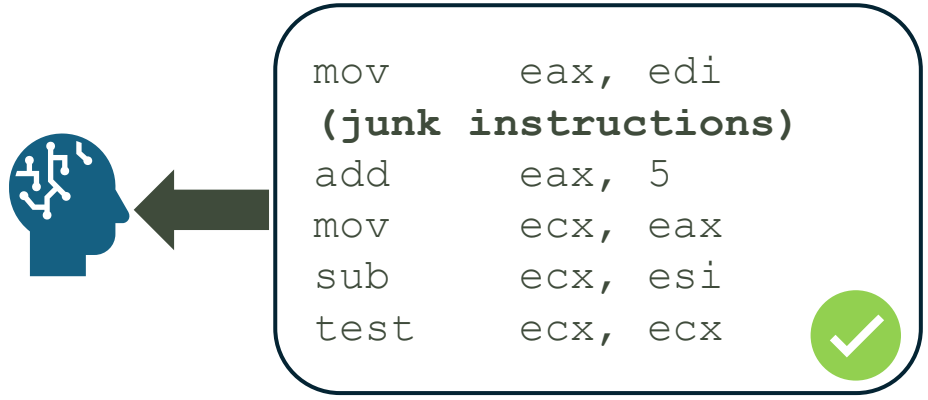


Transformation Budgets

Models often have an input limit

- We must consider this when adding additional instructions

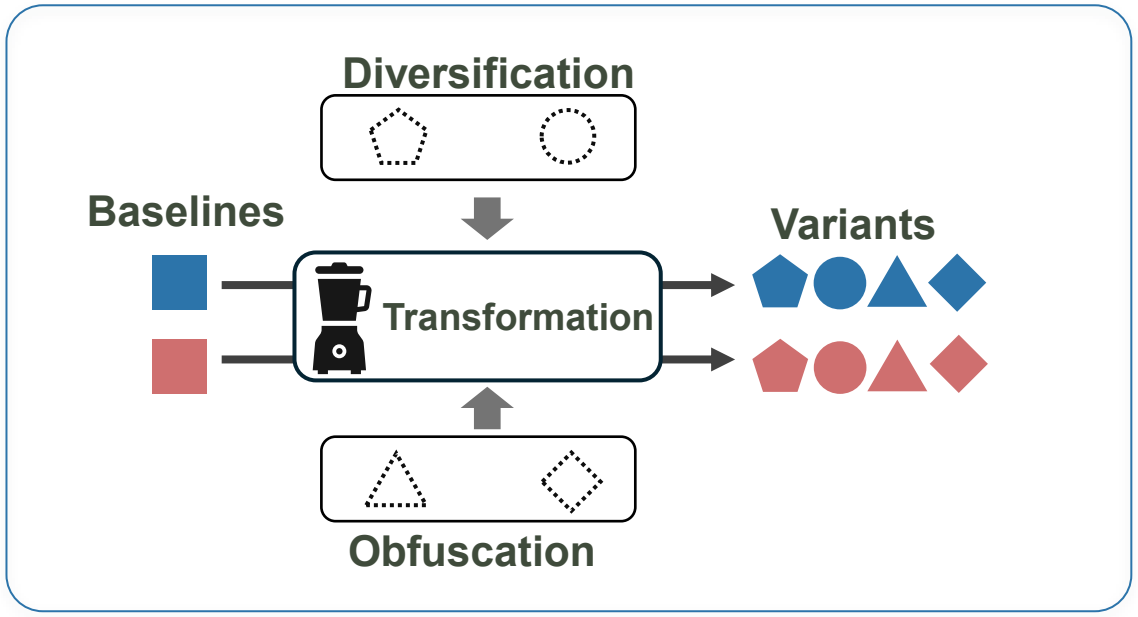


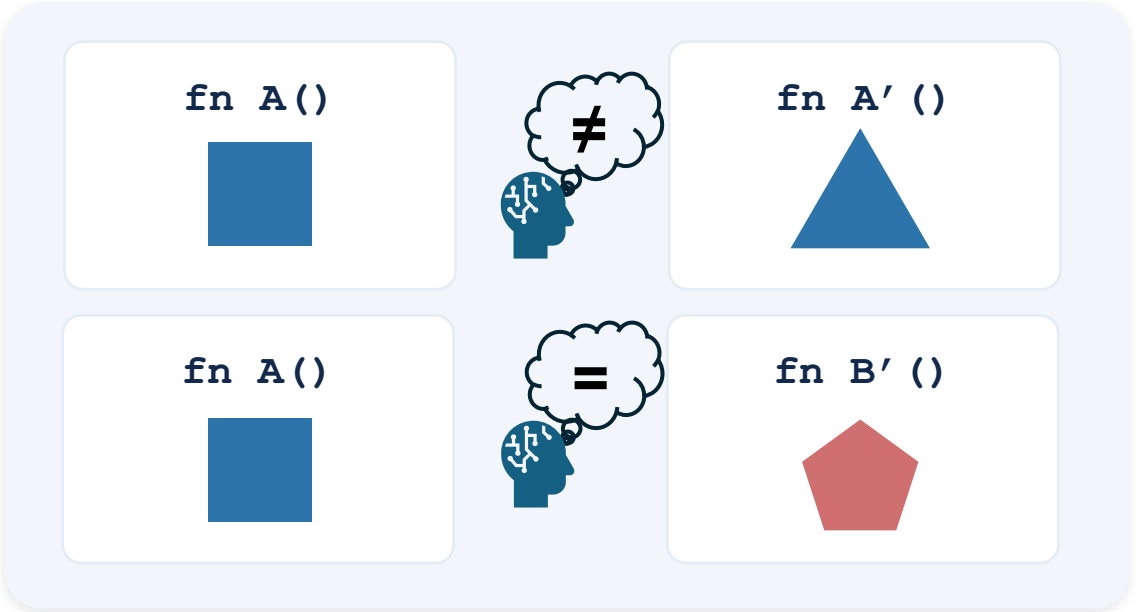


False Negative/Positive Perturbations

Making ***similar*** pairs look ***different*** vs Making ***different*** pairs look ***similar***

➔ Fundamentally different problems

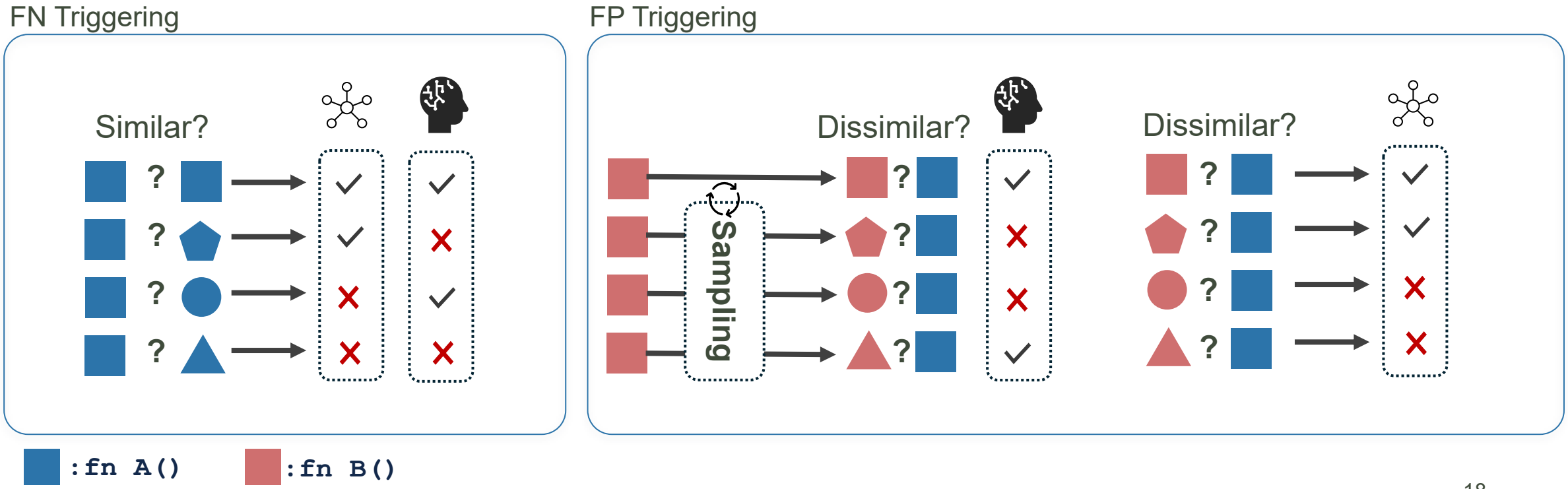




Experimental Setup

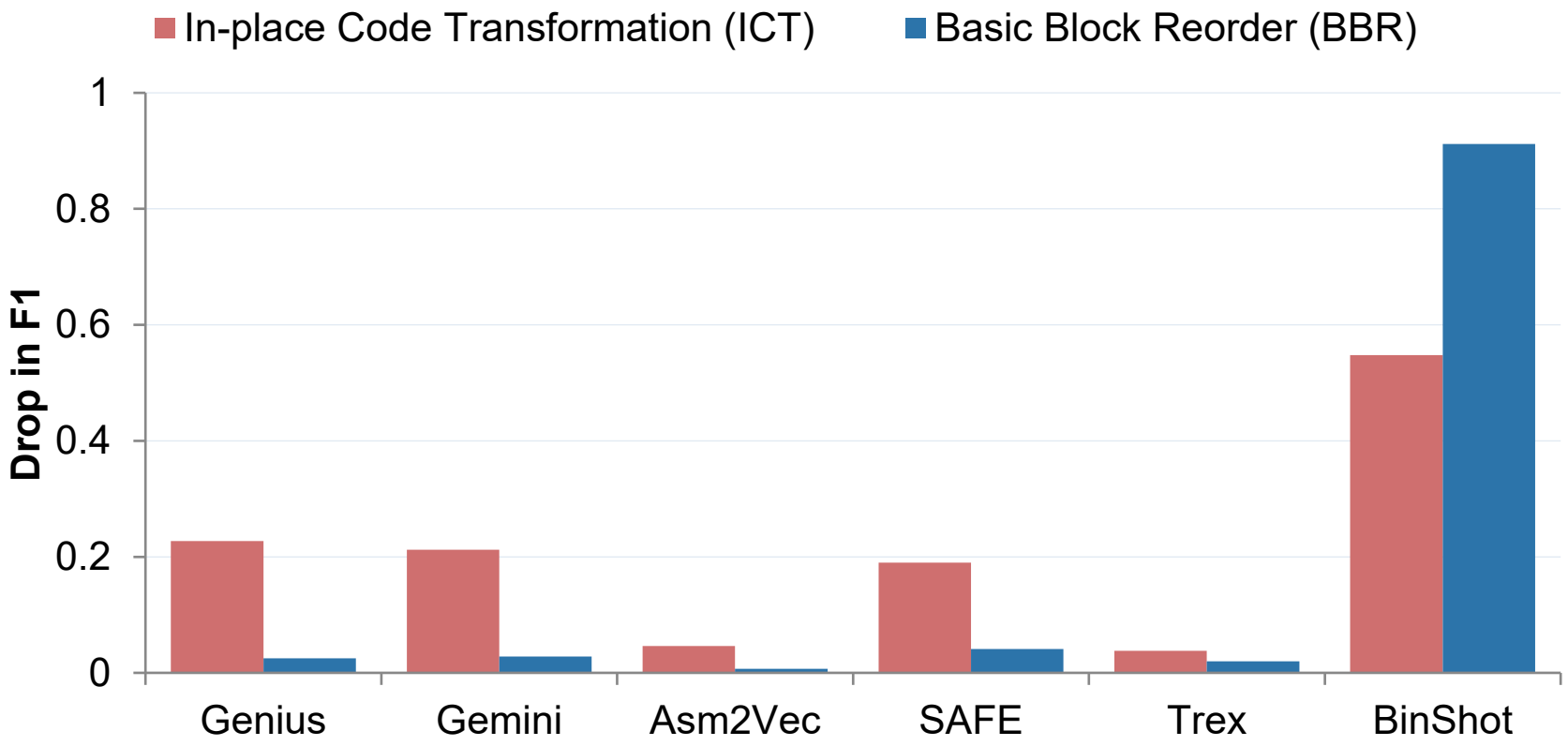
FN Triggering – all semantics-preserving transformations (all models)

FP Triggering – junk code insertion (target model: BinShot)



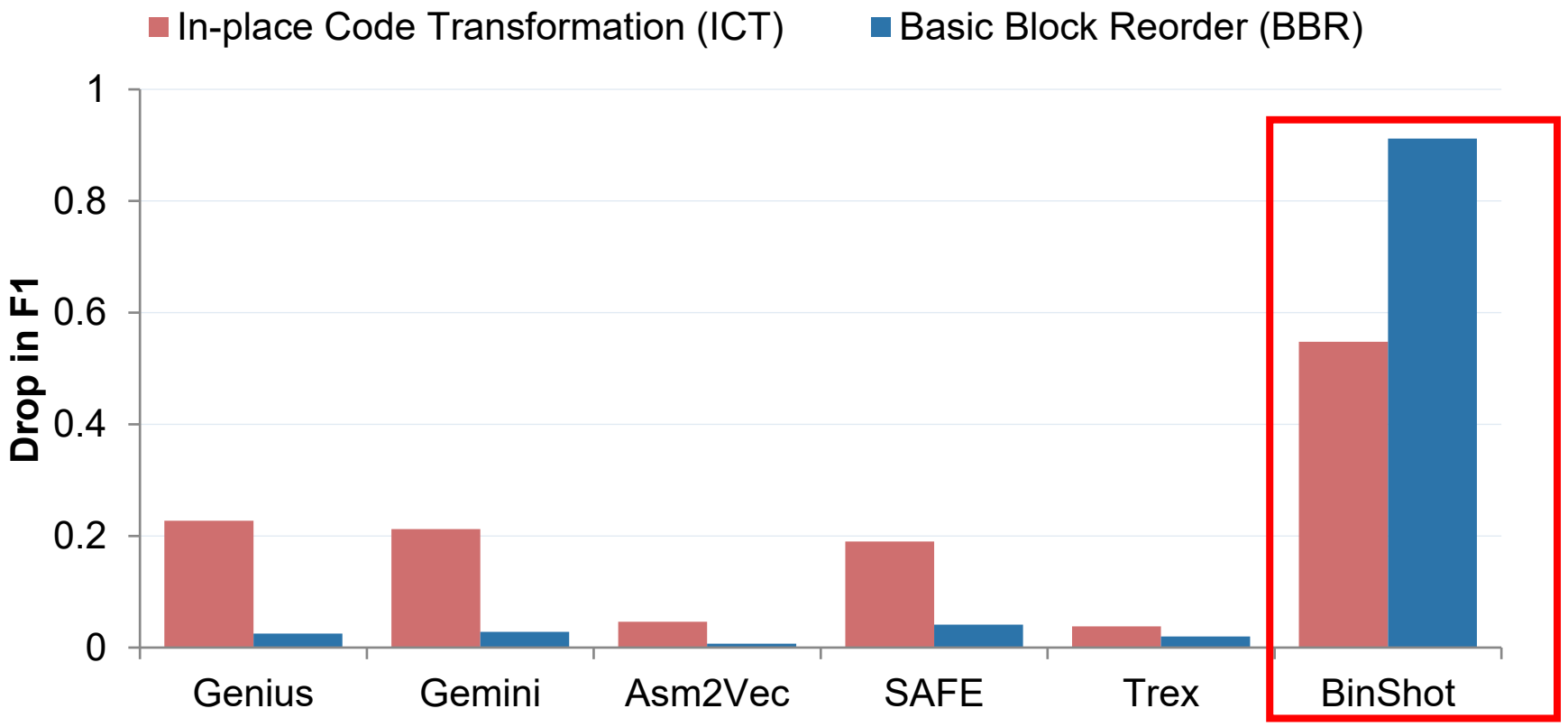
FN Triggering: Binary Diversification

Drop in F1 from two diversification transformations (smaller = robust)



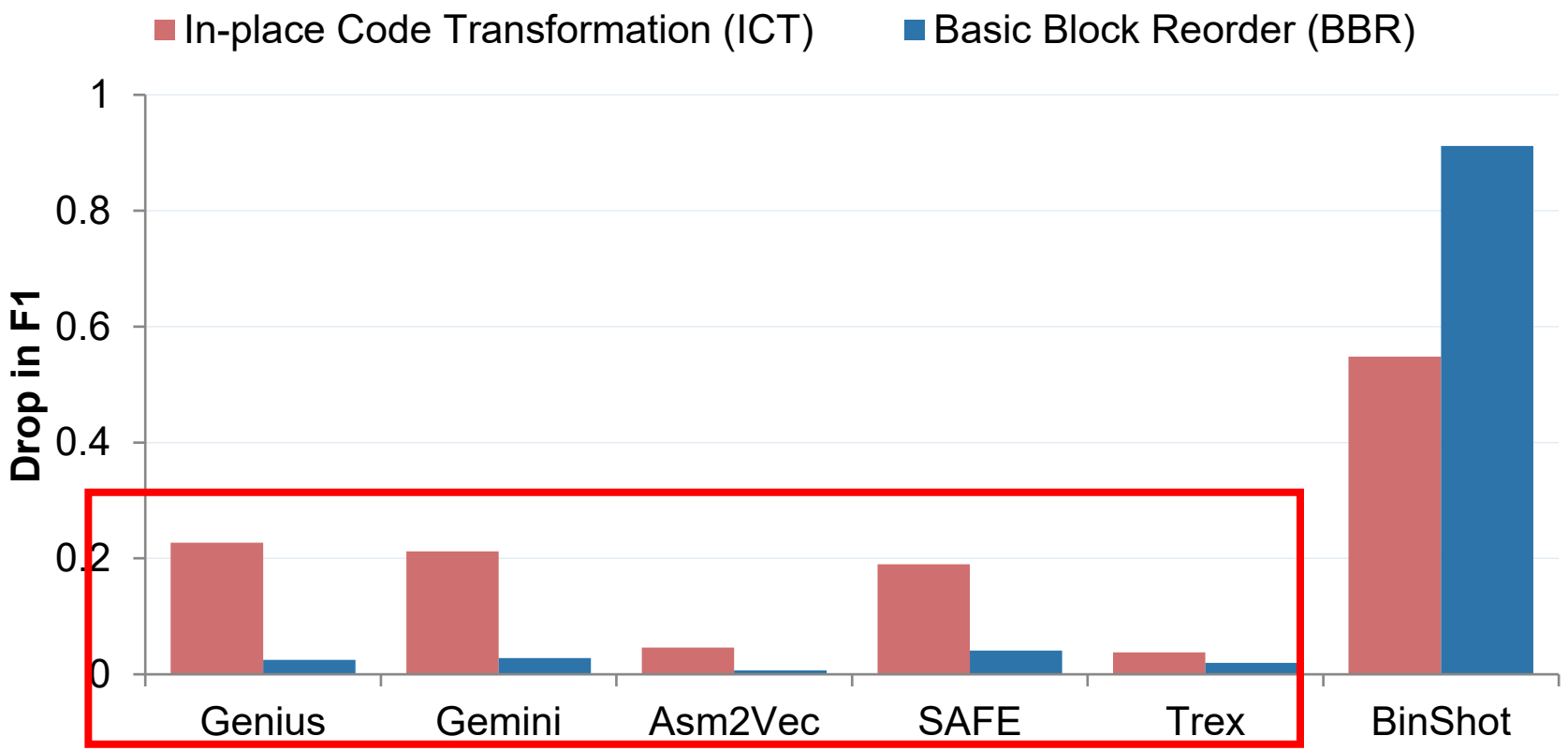
FN Triggering: Binary Diversification

BinShot leans on token order ➔ Significant impact on performance



FN Triggering: Binary Diversification

Robustness depends on how the model preprocesses data



FN Triggering: Obfuscation (O-LLVM)

O-LLVM Transformations

Substitution

Flattening

Bogus Control Flow

Apply All

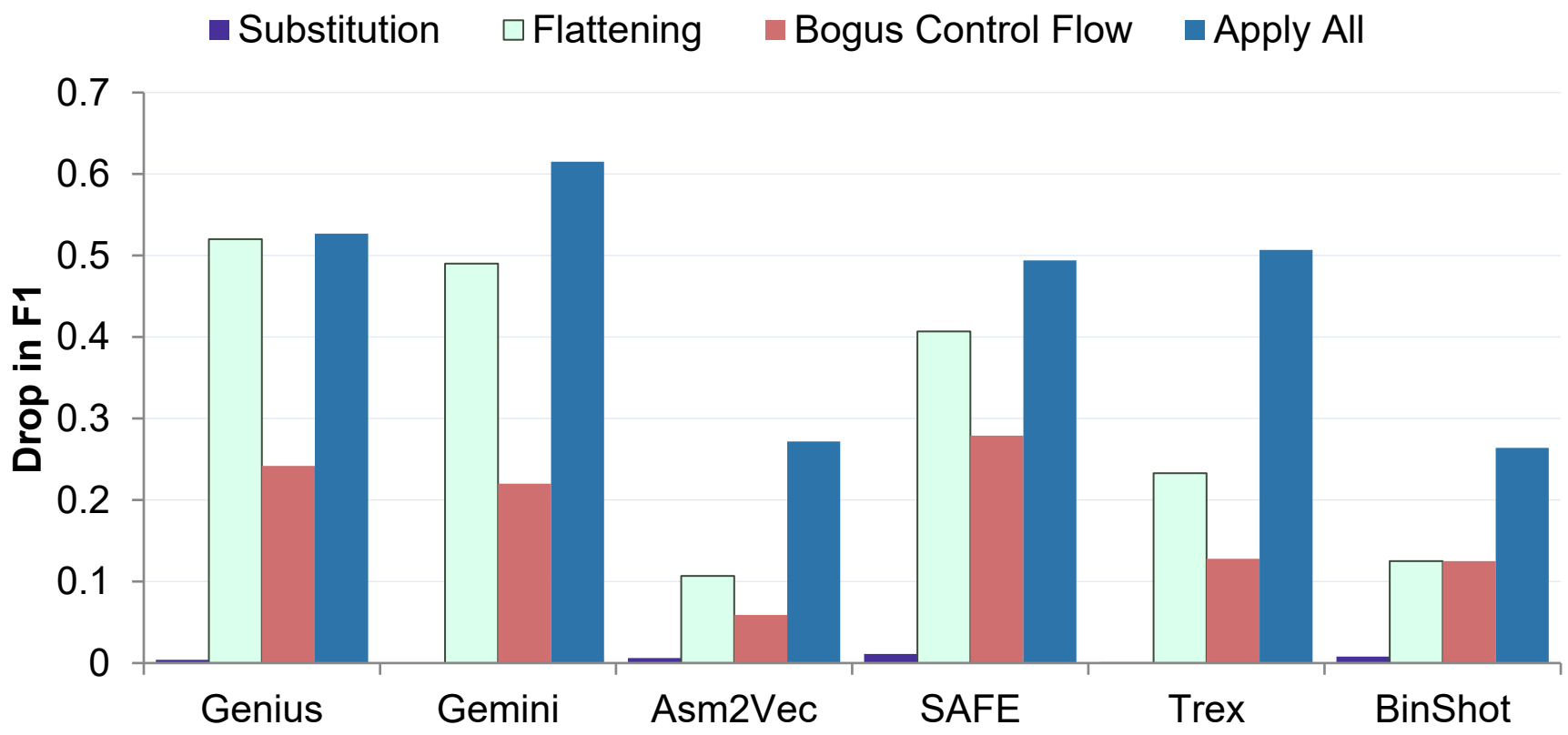
FN Triggering: Obfuscation (O-LLVM)

O-LLVM Transformations

Transformations that change the control flow graph

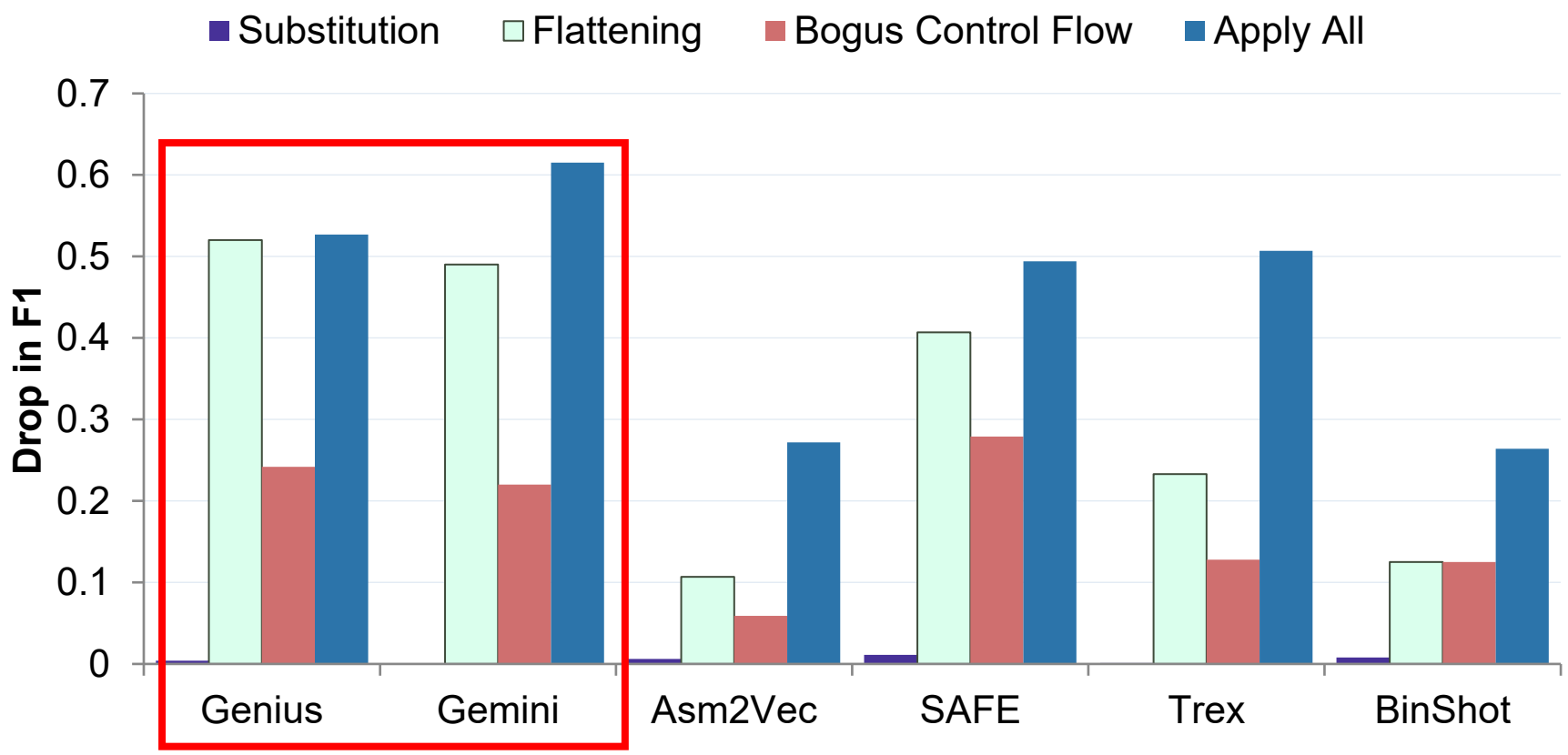
FN Triggering: Obfuscation (O-LLVM)

Drop in F1 compared to baseline results (smaller = robust)



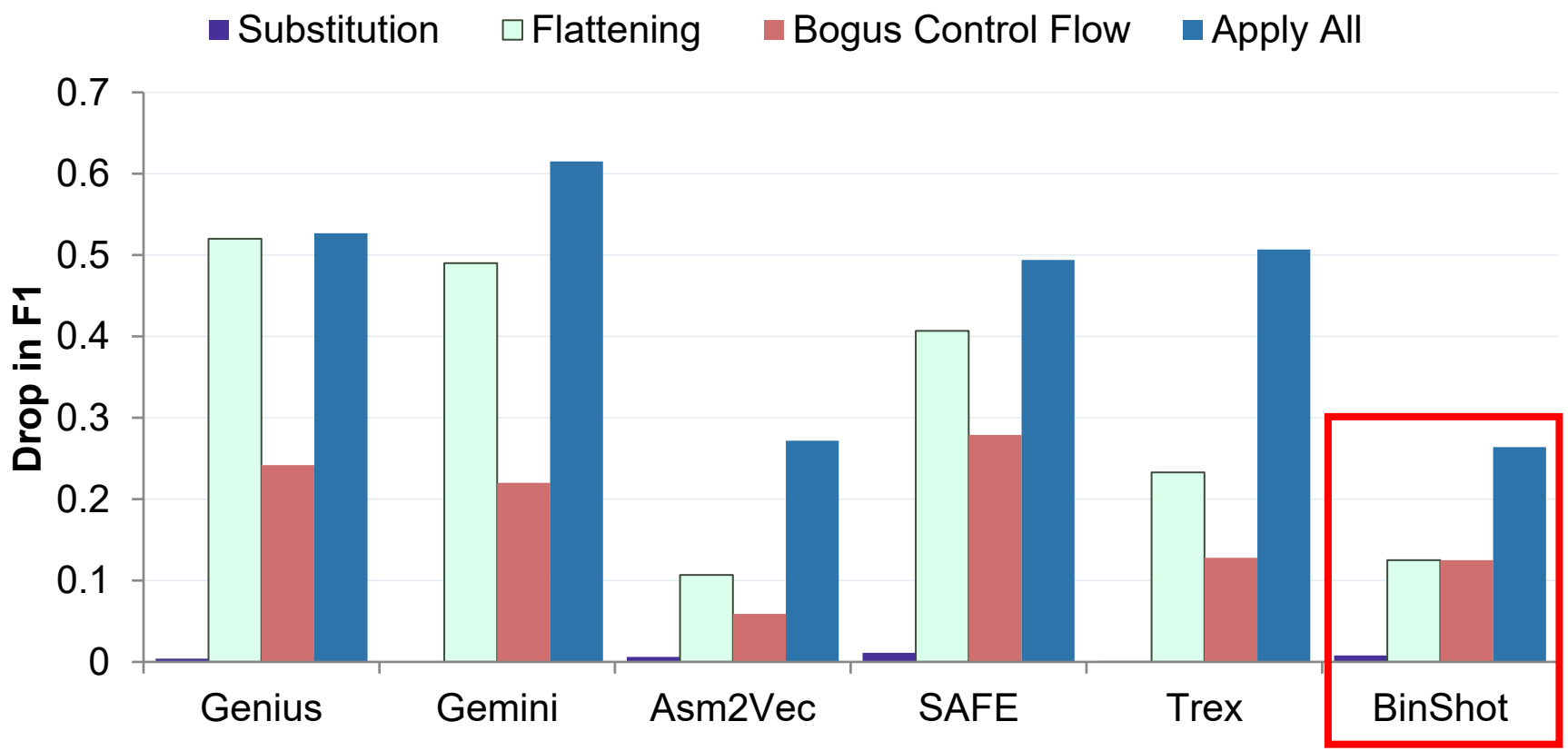
FN Triggering: Obfuscation (O-LLVM)

GNNs are significantly affected by transformations to CFG



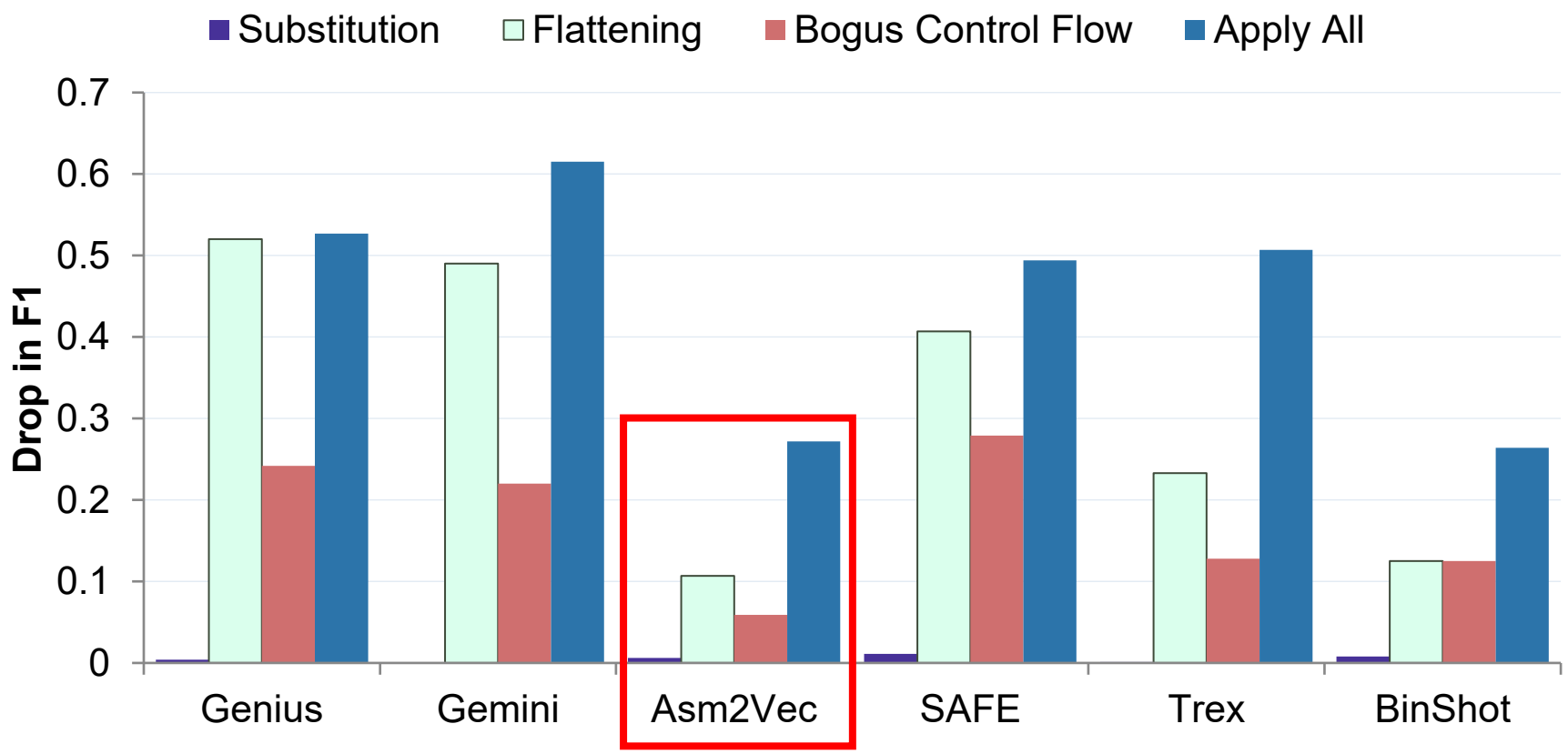
FN Triggering: Obfuscation (O-LLVM)

BinShot performs comparably well against O-LLVM

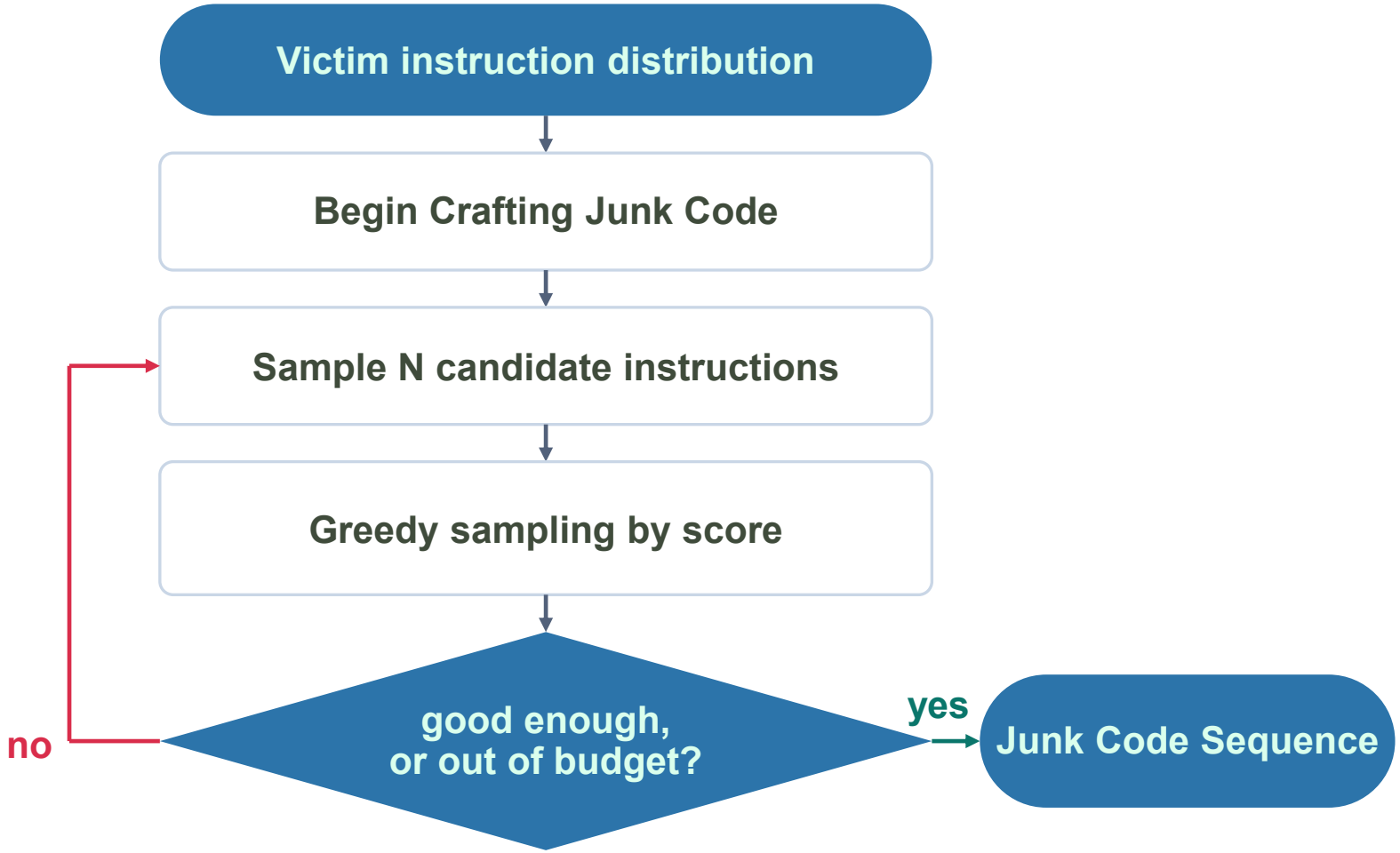


FN Triggering: Obfuscation (O-LLVM)

Asm2Vec does show low drop in F1, but it has lower baseline performance



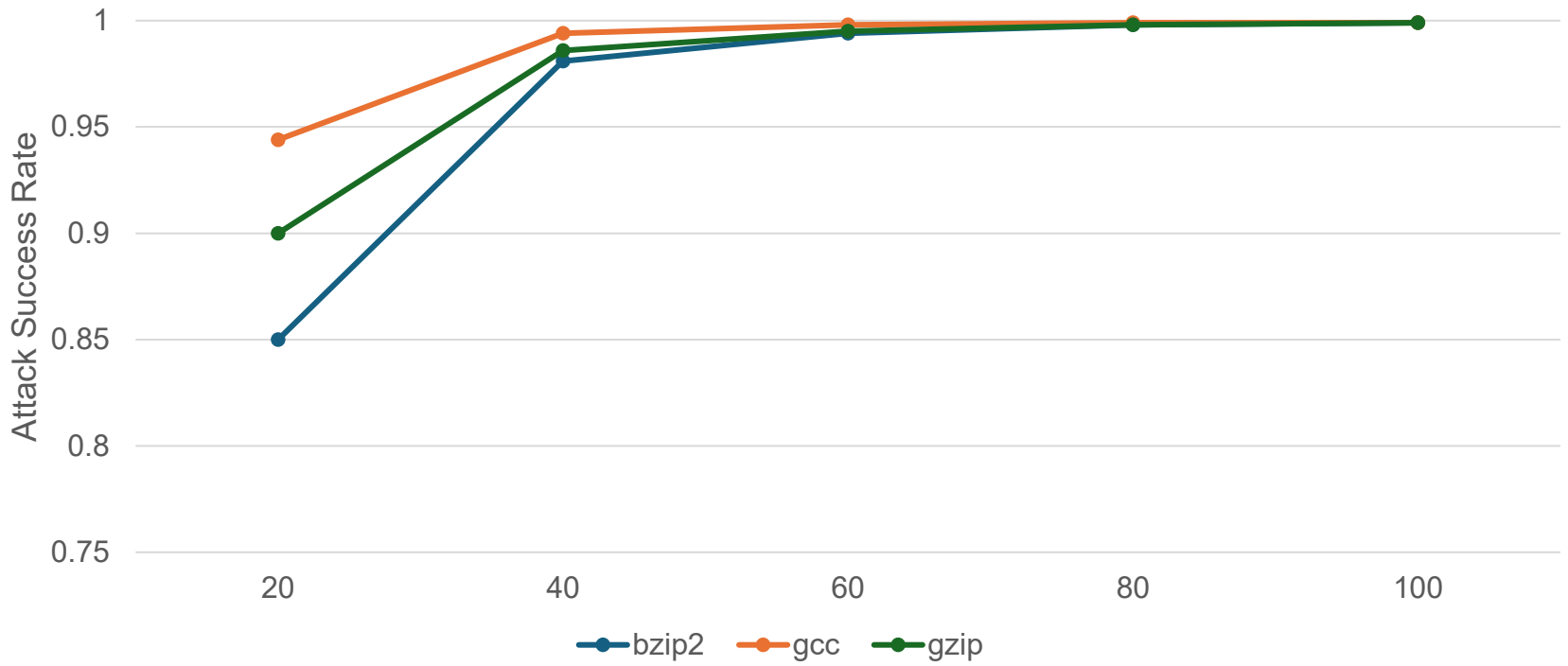
FP-Triggering Instruction Sampling



FP-Triggering Perturbation

Given a target function we add ***N*** (20~100) instructions to trick BinShot

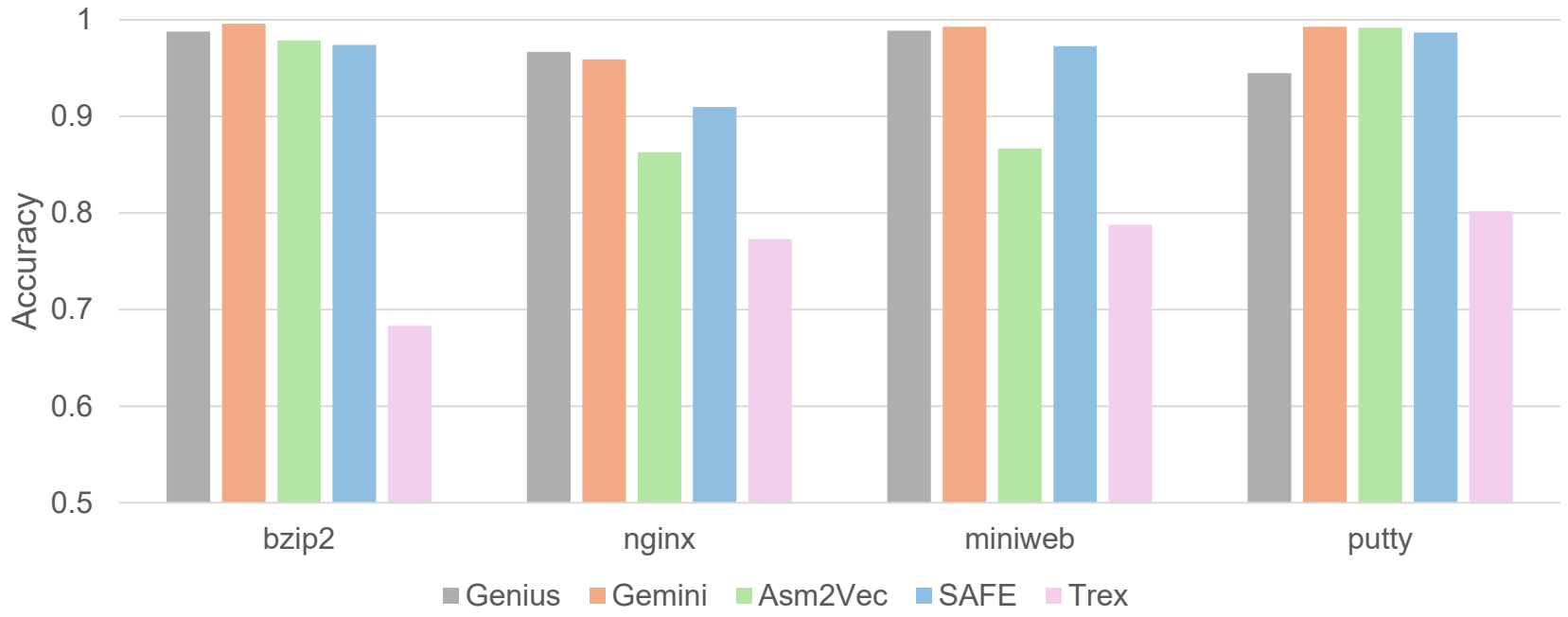
(ASR: smaller = robust)



FP-Triggering Perturbation: Transferability

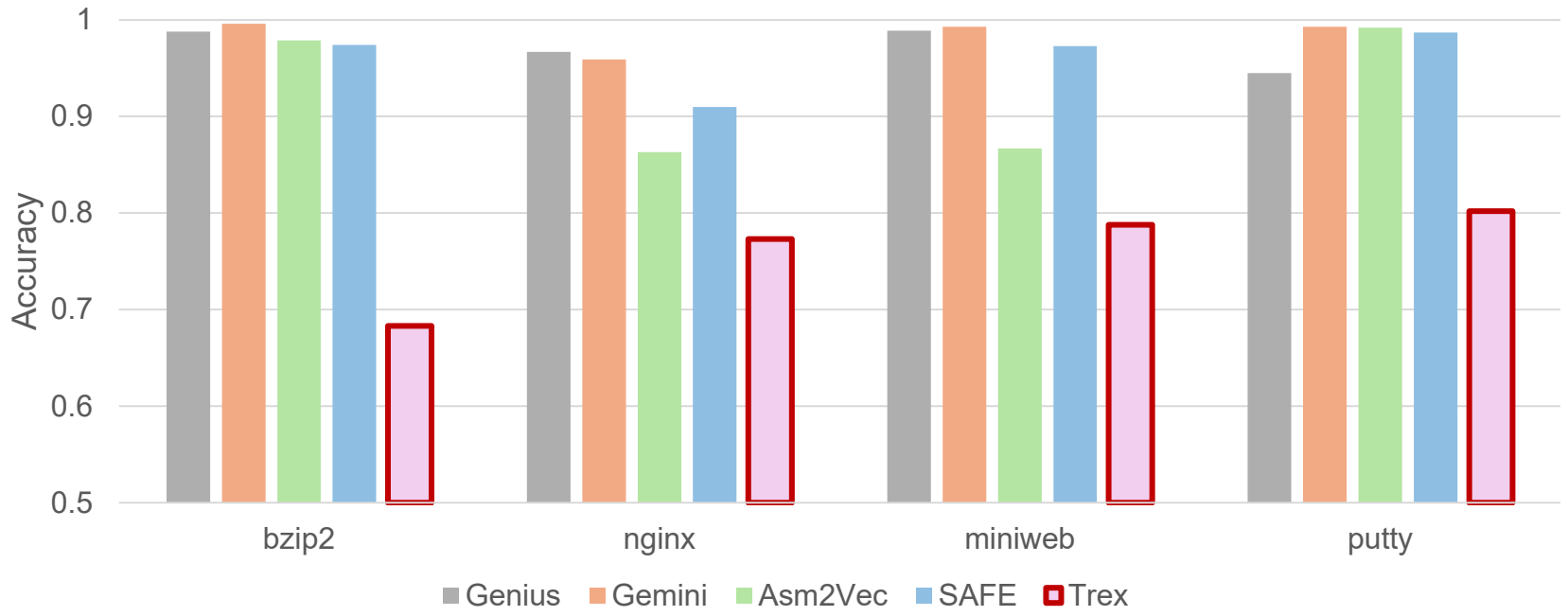
FP-Triggering results (e.g., BinShot). How well does it transfer?

(Accuracy: larger = robust)



FP-Triggering Perturbation: Transferability

Similar models (e.g, Trex) show notable transferability



Limitations and Future Work

- Transformations are not formally verified
- We do not explore all possible variants for a given transformation
- Packing, encryption, and commercial obfuscators are not covered
- Possible defenses (e.g., adversarial training) are not covered

Conclusion

- Model robustness significantly depends on the processing pipeline
  - Code preprocessing
  - Model architecture
  - Feature selection
- Effectiveness of semantics-preserving transformations are limited by
  - Inherent constraints of the model
  - Expressive capacity of semantically equivalent instructions within the binary
- Details on each transformation and results are in our paper!

Thank You

Feel free to ask questions via email ☺

jiyong423@g.skku.edu

Link to code

Link to paper

Explainable AI

XAI shows transformations move samples across the decision boundary