组会汇报

张文博

BinVulDet: Detecting vulnerability in binary program via decompiled pseudo code and BiLSTM-attention

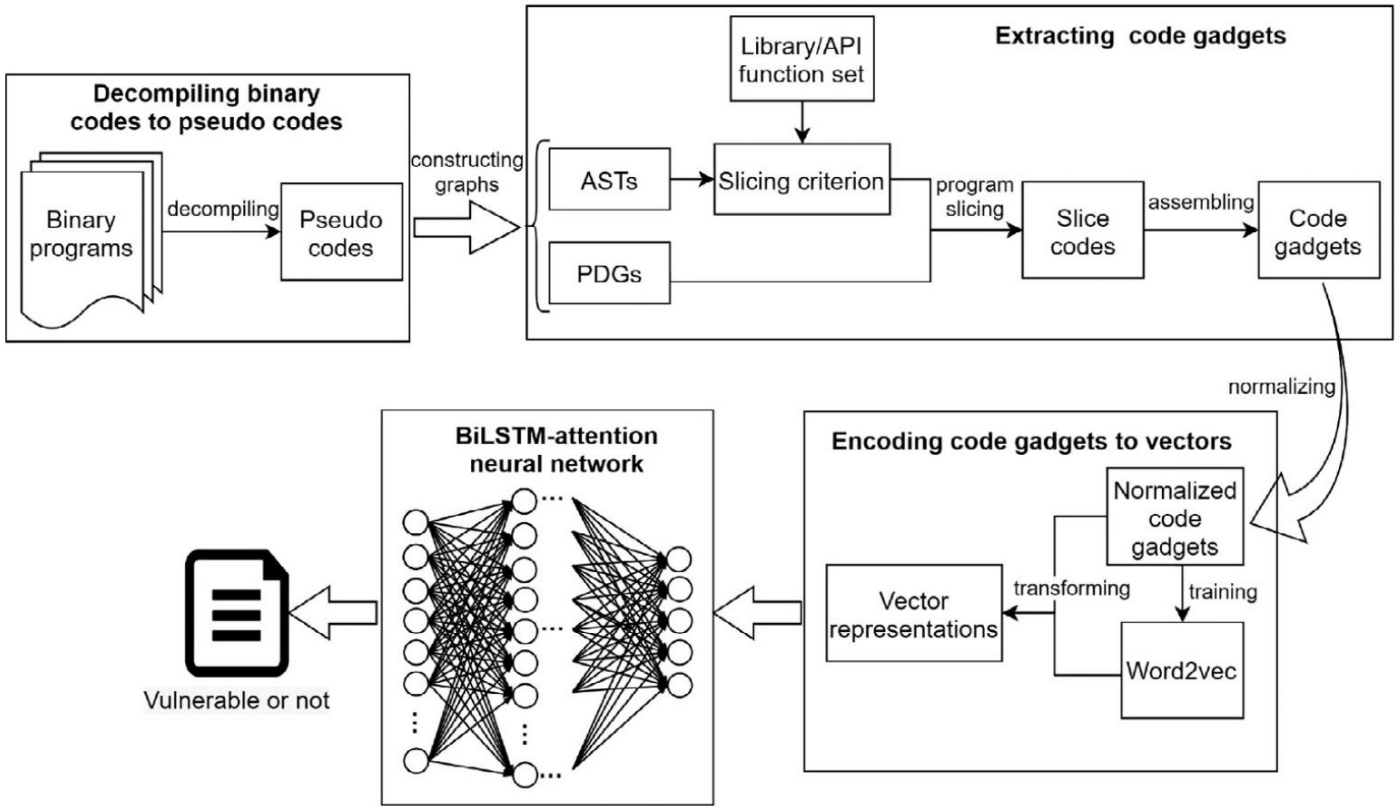


Fig. 1. Overview of BinVulDet.

实验结果

Table 5
Performance of each approach on the STV_CGD dataset.

System	FPR(%)	FNR(%)	A(%)	P (%)	F1(%)
Instruction2vec	7.43	9.11	91.88	89.33	90.10
VulDeePecker	11.69	13.17	87.66	84.49	85.13
HAN	4.51	8.41	93.90	93.29	92.43
HAN-BSVD	1.46	3.59	97.66	97.90	97.15
BinVulDet	1.61	0.18	98.93	97.48	98.62

Table 6
Performance of each approach on the IDA_CGD dataset.

System	FPR(%)	FNR(%)	A(%)	P(%)	F1(%)
Gemini	3.73	48.76	90.01	68.96	58.80
i2v_attention	3.92	34.96	91.76	72.83	68.71
SAFE	2.02	48.22	91.87	79.60	62.75
BinVulDet	0.46	1.59	99.11	99.25	98.83

Table 8
Evaluation results of each approach under the cross compile optimization level datasets.

Dataset	System	FPR(%)	FNR(%)	A(%)	P(%)	F1(%)
CCOL1_CGD	Gemini	4.55	32.18	89.97	78.65	72.84
	i2v_attention	4.21	27.30	91.22	81.04	76.64
	SAFE	2.61	34.34	90.90	86.62	74.70
	BinVulDet	1.68	3.95	97.53	96.83	96.44
CCOL2_CGD	Gemini	6.55	25.02	85.89	88.82	81.31
	i2v_attention	5.36	22.42	87.65	90.93	83.73
	SAFE	7.43	19.89	87.40	88.46	84.08
	BinVulDet	0.90	11.14	95.85	97.86	93.14

STV_CGD：只包含Juliet数据集CWE121（栈溢出）漏洞，GCC O0优化，IDA Pro 反编译

IDA_CGD：Juliet 数据集调用库/API 函数相关的 55 个 CWE，也是训练数据集，采用STV_CGD相同策略

数据规模是122484个代码片段，47022个正样本，75462个负样本

CCOL1_CGD：数据来源同IDA_CGD，使用GCC O0和GCC O3优化，IDA Pro反编译

CCOL2_CGD：数据来源同IDA_CGD，使用GCC O2和GCC O3优化，IDA Pro反编译

Code is not Natural Language: Unlock the Power of Semantics-Oriented Graph Representation for Binary Code Similarity Detection

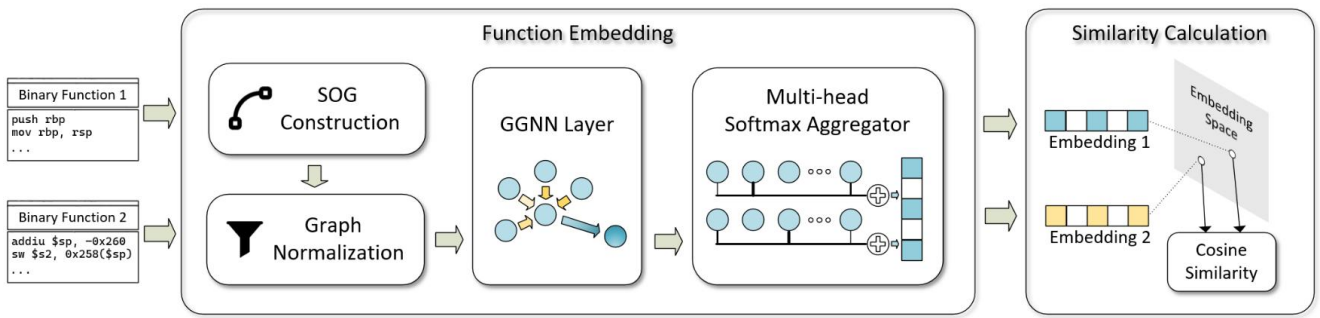
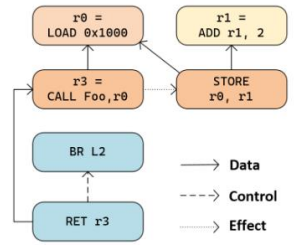


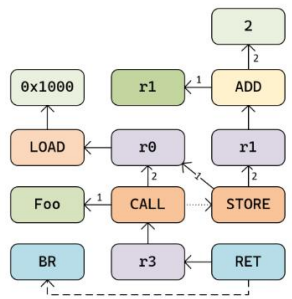
Figure 5: The overall framework of HermesSim.

```
1 L1:
2   r0 = LOAD 0×1000
3   r1 = ADD r1, 2
4   STORE r0, r1
5   r3 = CALL Foo, r0
6   BR L2
7 L2:
8   RET r3
```

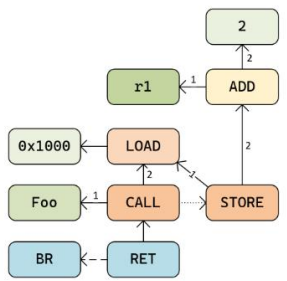
(a) Linear Representation in the toy IR



(b) ISCG: Instruction-based Semantics-Complete Graph



(c) TSCG: Token-based Semantics-Complete Graph



(d) SOG: Semantics-Oriented Graph

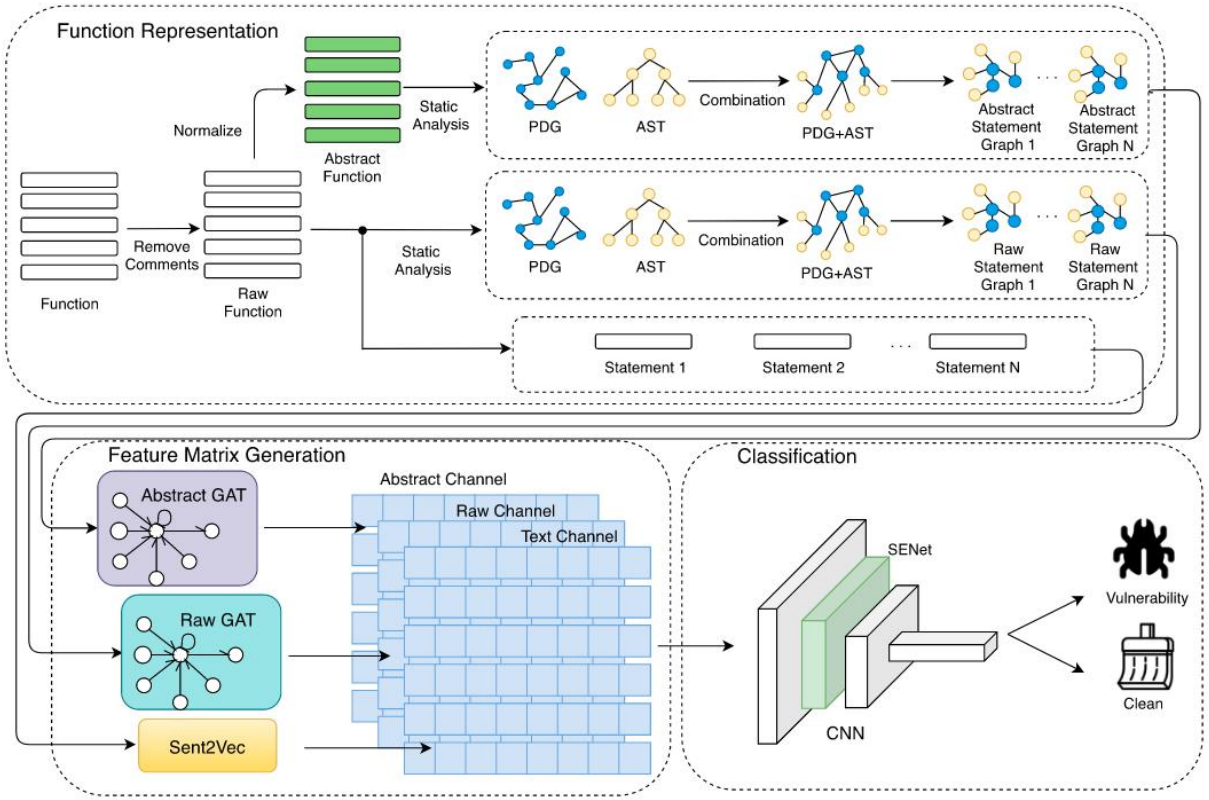
实验结果

实验数据集来源：
Andrea Marcelli, Mariano Graziano, Xabier Ugarte-Pedrero, and Yanick Fratantonio, Cisco Systems, Inc.; Mohamad Mansouri and Davide Balzarotti, EURECOM.**How Machine Learning Is Solving the Binary Function Similarity Problem**.USENIX Security '22.

该论文构建了一个通用二进制漏洞数据集，并在该数据集上对比了10个使用不同技术的分析方法。

[金山文档] How Machine Learning Is Solving the Binary Function Similarity Problem.pdf

Vulnerability Detection via Multiple-Graph-Based Code Representation



How Machine Learning Is Solving the Binary Function Similarity Problem

论文引用：Andrea Marcelli, Mariano Graziano, Xabier Ugarte-Pedrero, and Yanick Fratantonio, Cisco Systems, Inc.; Mohamad Mansouri and Davide Balzarotti, EURECOM.How Machine Learning Is Solving the Binary Function Similarity Problem.USENIX Security '22.

#代码相似性 #二进制代码分析

论文选取的用于比较的模型

字节模糊哈希

Catalog1: XORPD. FCatalog. HTTPS://WWW.XORPD.NET/PAGES/FCATALOG.HTM L.

该算法将函数字节作为输入并生成固定长度的签名，这是比较来自同一架构的函数的一种很有前途的方法。

程序控制流图模糊哈希

functionsimsearch: Thomas Dullien. Searching Statically-Linked Vulnerable Library Functions in Executable Code . HTTPS://GOOGLEPROJECTZERO.BLOG SPOT.COM/2018/12/SEARCHING-STATICALLY-LINKED-VULNERABL E.HTML.

该算法计算了一个模糊哈希，该模糊哈希结合了从 CFG、助记符和汇编代码中提取的立即值中提取的图基元（即小型连通、非同构、归纳子图）。由于基于 CFG 的功能，该方法可能是跨架构的。

综合CFG和GNN

Gemini: Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. 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, page 363–376, New York, NY, USA, 2017. Association for Computing Machinery.

Gemini 使用 GNN（structure2vec）来计算从函数 ACFG 开始的函数嵌入（即具有

基本块级属性的控制流图）。这种方法标志着一个里程碑，因为它是第一个利用具有 Siamese 架构的 GNN来学习函数相似性。

综合CFG、GNN、GMN

GMN: Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In International Conference on Machine Learning, pages 3835–3845. PMLR, 2019.
它提出了一种新颖的图匹配模型来计算图对之间的相似度。作者将功能相似性作为实际用例之一进行了探索。这种方法提出了两个来自机器学习社区的前沿模型，系统安全研究人员尚未研究过这两个模型

IR、数据流分析、神经网络

zeek: Noam Shalev and Nimrod Partush. Binary Similarity Detection using Machine Learning. In Proceedings of the 13th Workshop on Programming Languages and Analysis for Security - PLAS '18, pages 42–47, Toronto, Canada, 2018. ACM Press
zeek在基本块级别对提升代码 (VEX IR) 执行数据流分析（切片）并计算链。然后，训练两层全连接神经网络来学习跨架构相似性任务。这种方法是结合了中间表示、数据流分析和机器学习的最先进的建议。

汇编代码嵌入

asm2vec: Steven H. H. Ding, Benjamin C. M. Fung, and Philippe Charland. Asm2Vec: Boosting Static Representation Robustness for Binary Clone Search against Code Obfuscation and Compiler Optimization. In 2019 IEEE Symposium on Security and Privacy (SP), pages 472–489, San Francisco, CA, USA, May 2019. IEEE
asm2vec 引入了更精细的指令级拆分和嵌入结构，以克服汇编指令的词汇外 (OOV) 问题的局限性。这种方法是完全无监督的，并且在单一架构实验中取得了最先进的结果。

汇编代码嵌入以及自注意力编码器

safe: Luca Massarelli, Giuseppe Antonio Di Luna, Fabio Petroni, Leonardo Querzoni, and Roberto Baldoni. Safe: Self-Attentive Function Embeddings for Binary Similarity. In Proceedings of Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), 2019.

SAFE使用自注意力句子编码器学习跨架构函数嵌入。这种方法是 SEQ2SEQ 模型中 NLP 编码器的代表，与 ASM2VEC 相比，它是专门为学习跨架构相似性而设计的。

汇编代码嵌入、CFG、GNN

MASSARELLI ET AL., 2019.: LUCA MASSARELLI, GIUSEPPE A. DI LUNA, FABIO PETRONI, LEONARDO QUERZONI, AND ROBERTO BALDONI. INVESTIGATING GRAPH EMBEDDING NEURAL NETWORKS WITH UNSUPERVISED FEATURES EXTRACTION FOR BINARY ANALYSIS. IN PROCEEDINGS 2019 WORKSHOP ON BINARY ANALYSIS RESEARCH, SAN DIEGO, CA, 2019. INTERNET SOCIETY.
使用与GEMINI相同的STRUCTURE2VEC GNN，但它改变了块级特征，从手动设计的特征切换到无监督的特征。这种方法很有趣，因为它是 GEMINI 的演变，并且结合了指令级嵌入、基本块编码器和 GNN 的优点。

CODECMR/BINARYAI.

CODECMR: ZEPING YU, WENXIN ZHENG, JIAQI WANG, QIYI TANG, SEN NIE, AND SHI WU. CODECMR: CROSS-MODAL RETRIEVAL FOR FUNCTION-LEVEL BINARY SOURCE CODE MATCHING. ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 33, 2020.
该模型将中间表示与 NLP 编码器相结合以获得基本块嵌入，并使用 GNN 来获得图嵌入。两个 LSTM 对函数中的字符串和整数数据进行编码。函数嵌入是三者的串联，并且二进制模型是端到端训练的。

TREX

TREX: KEXIN PEI, ZHOU XUAN, JUNFENG YANG, SUMAN JANA, AND BAISHAKHI RAY. TREX: LEARNING EXECUTION SEMANTICS FROM MICRO-TRACES FOR BINARY SIMILARITY. ARXIV PREPRINT ARXIV:2012.08680, 2020.
TREX是最近基于分层变压器(HIERARCHICAL TRANSFORMER)和微迹线(MICRO-TRACES)的工作。本文带来了一个动态组件，可以提取函数轨迹，这对于学习函数语义至关重要。作者根据这些痕迹对 ML 模型进行预训练，并将学到的知识转移到匹配语义相似的函数。匹配阶段完全基于静态特征，而仅在预训练期间才需要生成微痕迹的仿真。这种跨架构解决方案构建在 TRANSFORMER 之上.

提供两个数据集

DATASET-1

训练集

- 由七个开源项目组成
  - clamav
  - curl
  - nmap
  - openssl
  - unrar
  - z3
  - zlib
- 含有24个不同的库。每个库都使用两个编译器系列（GCC 和 clang）进行编译，每个编译器系列有四个不同的版本，涵盖 2015 年到 2021 年的主要版本每个库都编译为三个不同的架构，X86-64、arm 和 mips，有 32 位和 64 位版本（总共 6 种架构组合），以及 5 个优化级别 O0、O1、O2、O3 和 OS。
- 禁用函数内联（function inlining）
- 数据集总共包含 5,489 个二进制文件，每个二进制项目平均有 228 个组合，总共有 2680 万个函数。
- 过滤了少于5个基本块的函数，约18.2m个
- 剩余的 860 万个函数是构建训练、验证和测试数据集的起点。

dataset-2

测试集

- 构建在最近发表的一篇论文trex 的作者发布的二进制文件之上。
- 从 13 个库中选择了 10 个库，以避免与 dataset-1 发生任何交叉：
  - binutils
  - coreutils
  - diffutils
  - findutils
  - gmp
  - imagemagick
  - libmicrohttpd
  - libtomcrypt
  - putty
  - sqlite
- 该数据集包含已针对 X86、X64、arm 32 位和 mips 32 位、4 个优化级别（O0、O1、O2、O3）和 GCC-7.5 编译的二进制文件。
- 该数据集的作用：
  - 在多样化的大型二进制文件集合上验证 dataset-1 模型的结果
  - 包括与最近的 trex 方法的比较

模型任务

xo

函数对有不同的优化，但编译器、编译器版本和架构相同。

xc

函数对具有不同的编译器、编译器版本和优化，但架构和位数相同。

xc+xb

函数对具有不同的编译器、编译器版本、优化和位数，但架构相同。

xa

函数对具有不同的体系结构和位数，但具有相同的编译器、编译器版本和优化。

xa+xo

函数对具有不同的体系结构、位数和优化，但编译器和编译器版本相同。

xm

函数对来自任意体系结构、位数、编译器、编译器版本和优化。
子数据集：

- xm-s：小规格函数（少于20个基本块）
- xm-m：中规格函数（多余20，少于100）
- xm-l：大规格函数（多余100）

说明

1. xo、xc、xc+xb，仅针对于单一架构技术
2. xa，适用于使用一致的编译器和编译器选项交叉编译的固件映像分析
3. xa+xo，支持dataset-2（仅使用一种编译器和编译器版本进行编译）
4. xm，最高难度，综合所有因素

评价维度

auc

ROC曲线线下面积auc
模型在所有可能的分类阈值上的性能的聚合度量

mrr and recall@k

平均倒数排名（mrr）和不同 K 阈值下的召回率（recall@K）。
排名度量对于评估那些需要通过大型数据库搜索候选函数的应用程序中的模型性能非常有用。

评估过程

	Description	Free variable				
		Arch	Bit	Comp	Opt	Ver
Catalog1	B + size 16	0.49	0.63	0.63	0.75	0.94
Catalog1	B + size 128	0.43	0.76	**0.85**	**0.92**	**0.99**
FSS	G	**0.81**	**0.89**	0.68	0.74	0.87
FSS	G + M	0.66	0.88	0.78	0.83	0.97
FSS	G + M + I	0.67	0.88	0.77	0.82	0.97
FSS	w(G + M + I)	0.75	0.83	0.67	0.74	0.82

Table 2: Comparison of Catalog1 and FunctionSimSearch (FSS) on Dataset-1.

						XM					XC+XB	
	Description	XC	XC+XB	XA	XM	small	medium	large	MRR10	Recall@1	MRR10	Recall@1
Catalog1	B + size 16	0.66	0.60	0.48	0.54	0.54	0.53	0.54	0.08	0.07	0.25	0.23
Catalog1	B + size 128	**0.73**	0.66	0.43	0.55	0.54	0.55	0.58	0.12	0.09	**0.31**	**0.27**
FSS	G	0.72	**0.72**	**0.69**	**0.70**	**0.70**	**0.71**	**0.77**	**0.26**	**0.20**	0.29	0.23
FSS	G + M	**0.73**	0.71	0.58	0.65	0.64	0.66	0.70	0.17	0.13	0.29	0.24
FSS	G + M + I	**0.73**	0.70	0.58	0.65	0.64	0.66	0.71	0.15	0.09	0.28	0.23
FSS	*w* (G + M + I)	0.69	0.69	0.65	0.67	0.66	0.68	0.72	0.21	0.16	0.23	0.17

Table 3: Comparison of machine-learning models on Dataset-1.

						XM				
	Description	XC	XC+XB	XA	XM	small	medium	large	MRR10	Recall@1
[67] Zeek (direct comparison)	Strands	0.84	0.85	0.84	0.84	0.85	0.83	**0.87**	0.28	0.13
[40] GMN (direct comparison)	CFG + BoW opc 200	0.85	0.86	**0.86**	0.86	**0.89**	0.82	0.79	**0.53**	**0.45**
[40] GMN (direct comparison)	CFG + No features	**0.86**	**0.87**	**0.86**	**0.87**	0.88	**0.85**	0.84	0.43	0.33
[40] GNN	CFG + BoW opc 200	**0.86**	**0.87**	**0.86**	**0.87**	**0.89**	0.84	0.76	0.52	0.44
[40] GNN	CFG + No features	0.82	0.83	0.82	0.82	0.85	0.80	0.76	0.37	0.29
[76] GNN (s2v)	CFG + BoW opc 200	0.81	0.82	0.78	0.81	0.82	0.78	0.74	0.36	0.26
[76] GNN (s2v)	CFG + manual	0.81	0.82	0.80	0.81	0.84	0.77	0.79	0.36	0.28
[76] GNN (s2v)	CFG + No features	0.69	0.70	0.69	0.70	0.70	0.69	0.75	0.12	0.07
[45] w2v + AVG + GNN (s2v)	CFG + N. asm 150	0.79	0.79	0.74	0.77	0.78	0.75	0.73	0.24	0.16
[45] w2v + wAVG + GNN (s2v)	CFG + N. asm 150	0.79	0.79	0.76	0.77	0.78	0.76	0.76	0.29	0.20
[45] w2v + RNN + GNN (s2v)	CFG + N. asm 150	0.79	0.80	0.79	0.80	0.82	0.77	0.80	0.27	0.17
[49] w2v + SAFE	N. asm 150	0.80	0.81	0.80	0.81	0.83	0.77	0.77	0.17	0.07
[49] w2v + SAFE	N. asm 250	0.82	0.83	0.82	0.83	0.84	0.81	0.82	0.22	0.09
[49] w2v + SAFE + trainable	N. asm 150	0.80	0.81	0.80	0.81	0.83	0.76	0.74	0.29	0.16
[49] rand + SAFE + trainable	N. asm 150	0.79	0.80	0.79	0.80	0.83	0.75	0.74	0.28	0.17
[14] Asm2Vec	10 CFG random walks	0.77	0.69	0.60	0.65	0.63	0.70	0.78	0.12	0.07
[38] PV-DM	10 CFG random walks	0.77	0.70	0.50	0.62	0.63	0.62	0.61	0.11	0.08
[38] PV-DBOW	10 CFG random walks	0.78	0.70	0.50	0.63	0.63	0.62	0.61	0.11	0.09

Table 4: Comparison of fuzzy hashing and machine-learning models on Dataset-2

		AUC			MRR10			Recall@1			Testing time (s)		
Model name	Description	XO	XA	XA+XO	XO	XA	XA+XO	XO	XA	XA+XO	Feat	Inf	Tot 100
[67] Zeek (direct comparison)	Strands	0.92	0.94	0.91	0.42	0.45	0.36	0.28	0.31	0.21	7225.41	67.00	9.92
[40] GMN (direct comparison)	CFG + BoW opc 200	**0.97**	**0.98**	**0.96**	**0.75**	**0.84**	**0.71**	**0.66**	**0.77**	**0.61**	1093.68	1005.00	1.83
[40] GMN (direct comparison)	CFG + No features	0.93	0.97	0.95	0.61	0.76	0.67	0.51	0.68	0.59	978.15	876.00	1.63
[40] GNN	CFG + BoW opc 200	0.95	0.97	0.95	0.67	0.79	0.67	0.57	0.73	0.57	1093.68	116.52	1.66
[40] GNN	CFG + No features	0.91	0.96	0.93	0.54	0.71	0.59	0.44	0.62	0.49	978.15	100.34	1.48
[76] GNN (s2v)	CFG + BoW opc 200	0.94	0.95	0.93	0.58	0.57	0.58	0.48	0.42	0.47	1093.68	118.59	1.66
[76] GNN (s2v)	CFG + Gemini	0.93	0.96	0.93	0.57	0.74	0.57	0.47	0.64	0.49	5139.91	98.40	7.18
[76] GNN (s2v)	CFG + No features	0.75	0.79	0.77	0.18	0.20	0.23	0.12	0.13	0.16	978.15	40.87	1.40
[45] w2v + AVG + GNN (s2v)	CFG + N. asm 150	0.90	0.88	0.87	0.46	0.31	0.42	0.38	0.18	0.33	1070.01	258.95	1.82
[45] w2v + wAVG + GNN (s2v)	CFG + N. asm 150	0.87	0.87	0.85	0.37	0.29	0.36	0.29	0.17	0.27	1070.01	253.72	1.81
[45] w2v + RNN + GNN (s2v)	CFG + N. asm 150	0.88	0.90	0.88	0.32	0.35	0.35	0.19	0.18	0.23	1070.01	685.50	2.41
[49] w2v + SAFE	N. asm 150	0.88	0.90	0.88	0.27	0.30	0.31	0.14	0.18	0.20	1031.23	33.33	1.46
[49] w2v + SAFE	N. asm 250	0.86	0.88	0.87	0.28	0.32	0.28	0.16	0.19	0.19	1031.23	33.33	1.46
[49] w2v + SAFE + trainable	N. asm 150	0.91	0.93	0.91	0.40	0.43	0.37	0.26	0.25	0.23	1031.23	33.57	1.46
[49] rand + SAFE + trainable	N. asm 150	0.90	0.91	0.90	0.28	0.33	0.31	0.14	0.17	0.21	1031.23	33.81	1.46
[14] Asm2Vec	Rand walks asm	0.94	0.69	0.75	0.60	0.07	0.22	0.49	0.02	0.18	978.15	5235.00	8.51
[38] PV-DM	Rand walks asm	0.94	0.66	0.72	0.64	0.08	0.23	0.51	0.05	0.19	978.15	5239.00	8.52
[38] PV-DBOW	Rand walks asm	0.94	0.66	0.72	0.63	0.07	0.23	0.50	0.03	0.20	978.15	3004.00	5.46
[60] Trex	512 Tokens	0.94	0.94	0.94	0.61	0.50	0.53	0.50	0.38	0.46	1493.58	1365.89	3.92
[74] Catalog_1	size 16	0.72	0.50	0.55	0.43	0.06	0.14	0.38	0.06	0.14	654.70	0.00	0.90
[74] Catalog_1	size 128	0.86	0.48	0.57	0.50	0.07	0.17	0.42	0.06	0.14	823.47	0.00	1.13
[18] FSS	G	0.77	0.81	0.77	0.26	0.35	0.32	0.18	0.26	0.26	1903.46	466.07	3.25
[18] FSS	G + M	0.79	0.68	0.69	0.29	0.15	0.21	0.23	0.09	0.15	1903.46	466.07	3.25
[18] FSS	G + M + I	0.80	0.68	0.69	0.30	0.16	0.20	0.23	0.10	0.14	1903.46	466.07	3.25
[18] FSS	w(G + M + I)	0.83	0.80	0.78	0.43	0.30	0.36	0.36	0.23	0.29	1903.46	466.07	3.25