Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

二进制函数相似性分析

硕士研究生 沈宇辉

2022年11月27日

内容提要

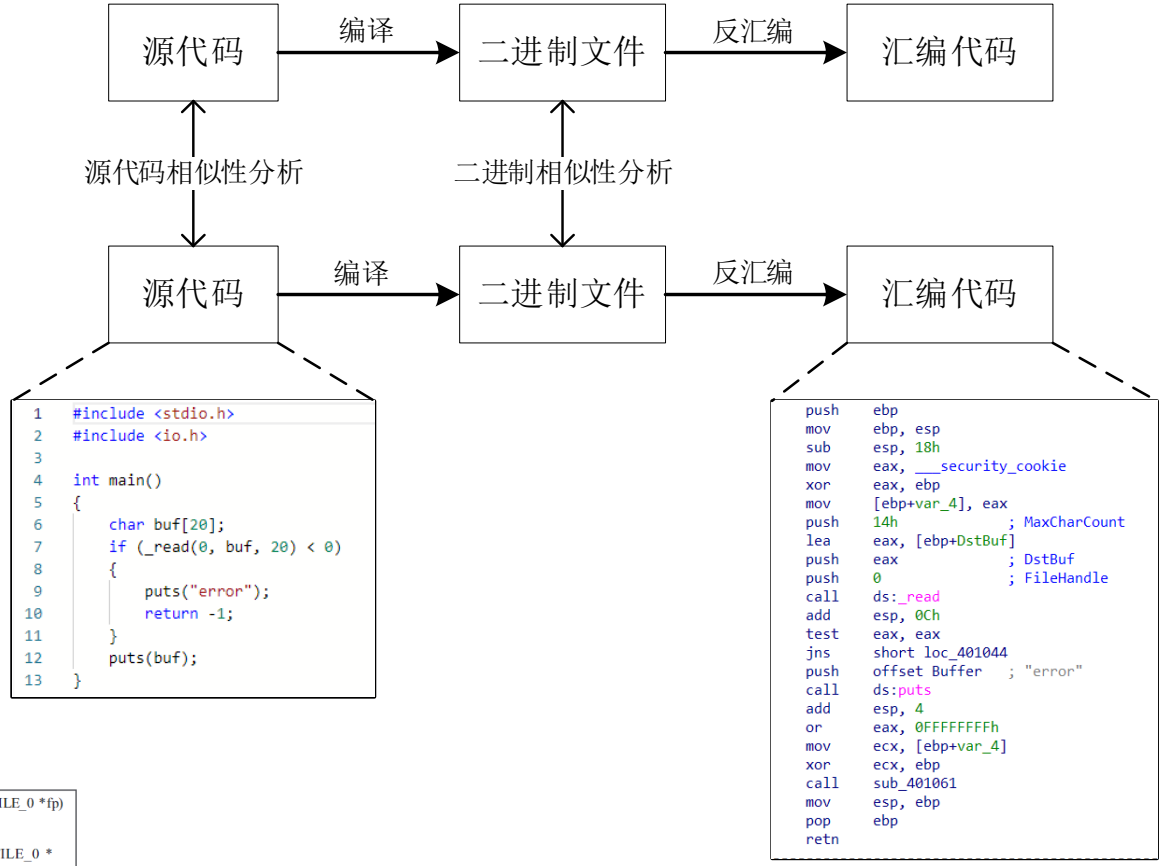
- 背景简介
- 基本概念
- 算法原理
- 总结
- 参考文献

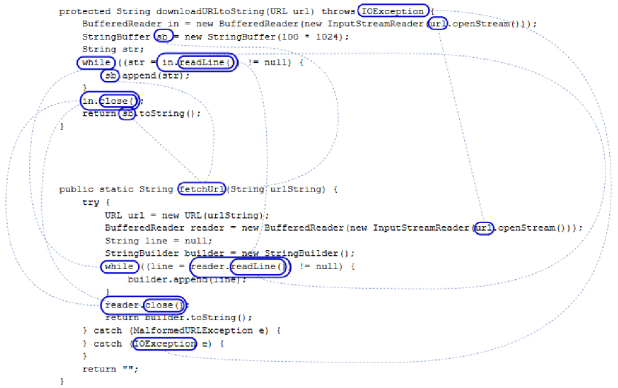
背景简介

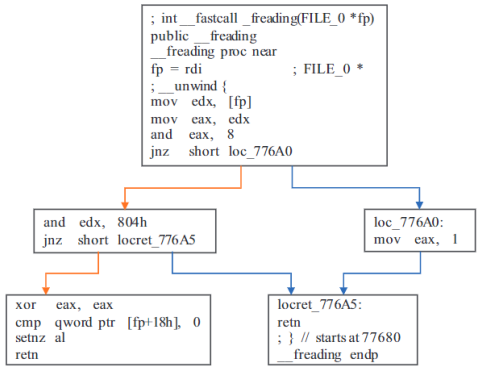
- 预期收获
  - 了解二进制相似性分析的发展历史
  - 理解二进制相似性分析的基本方法
  - 理解结构特征、语义特征的概念及相关原理

背景简介

- 二进制相似度分析
  - 分析对比二进制程序片段间的异同
- 与“代码克隆检测”的差异
  - 程序源代码无法获得
  - 编译过程带来信息损失
    - 函数边界模糊
    - 函数名、变量名不可知

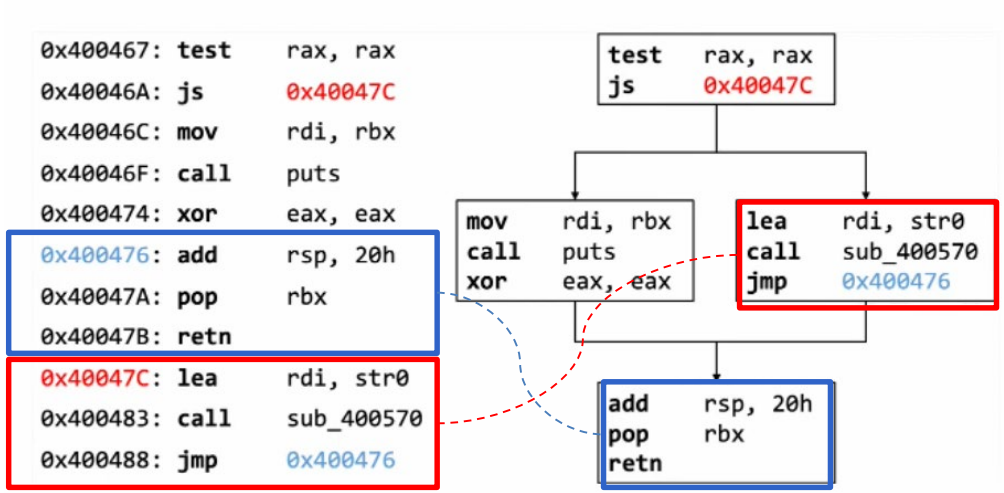






基本概念

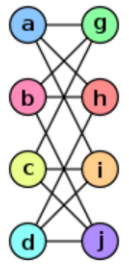
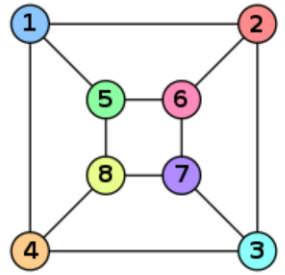
- 控制流图
  - 利用数学中图的表示方式，表示计算机程序执行过程中所经过的所有路径
  - 二进制程序的重要表征形式之一
- 基本块
  - 不包含分支的一段语句序列
  - 控制流图中的节点
- 跳转关系
  - 控制流图中的边
  - 由源基本块指向目标基本块



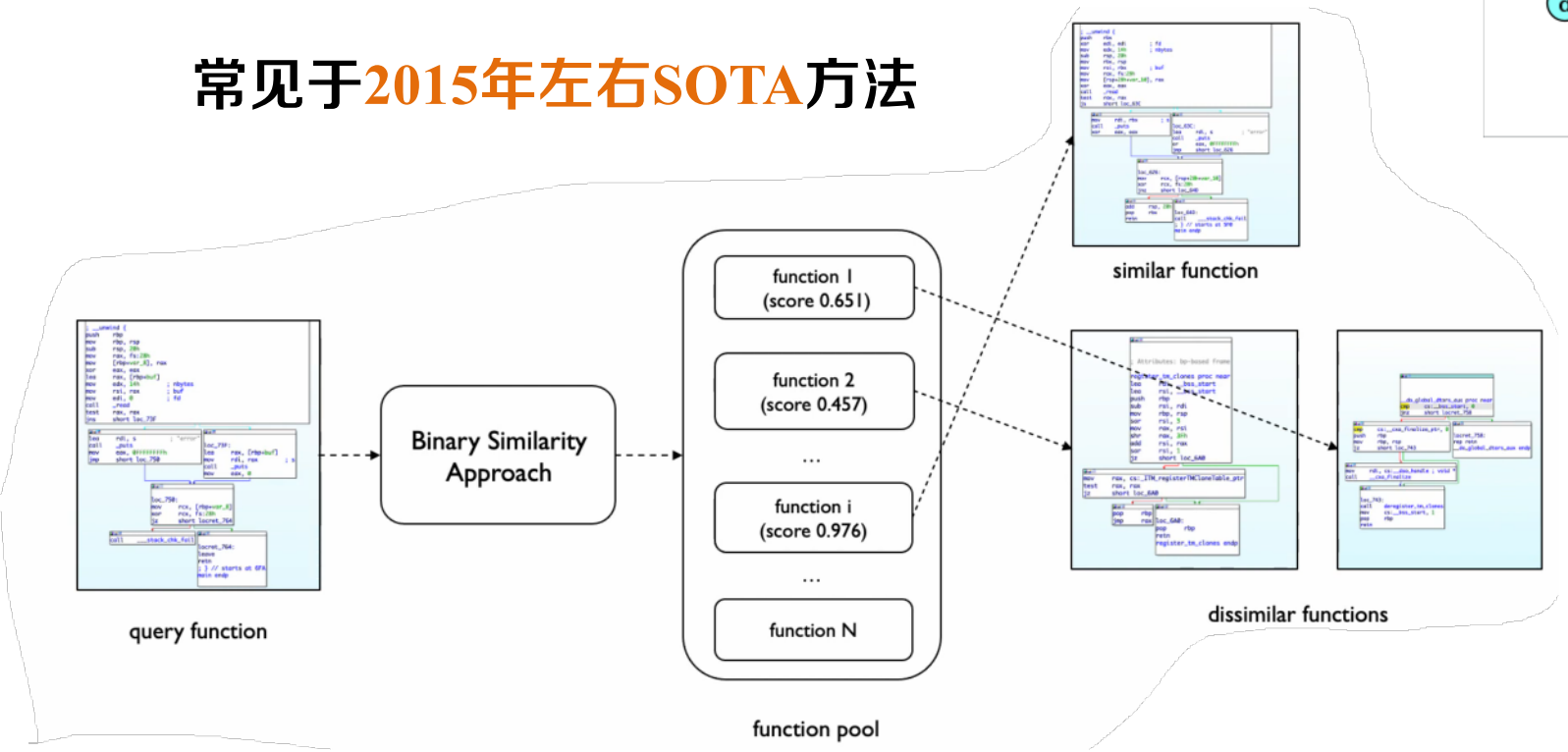
控制流图——二进制相似性分析的根基

基本概念

- 二进制相似性分析转化为同构图分析问题
  - 判断两个控制流图是否有近似等价关系
  - 两两计算控制流图相似度得分
  - 通过设定阈值/排序获得相似度分析结果

图G	图H	从图G到图H的同构映射σ
		$\sigma(a)=1$ $\sigma(b)=6$ $\sigma(c)=8$ $\sigma(d)=3$ $\sigma(g)=5$ $\sigma(h)=2$ $\sigma(i)=4$ $\sigma(j)=7$

常见于2015年左右SOTA方法



需求驱动创新：

多次一对一同构图分析效率过低？

基本概念

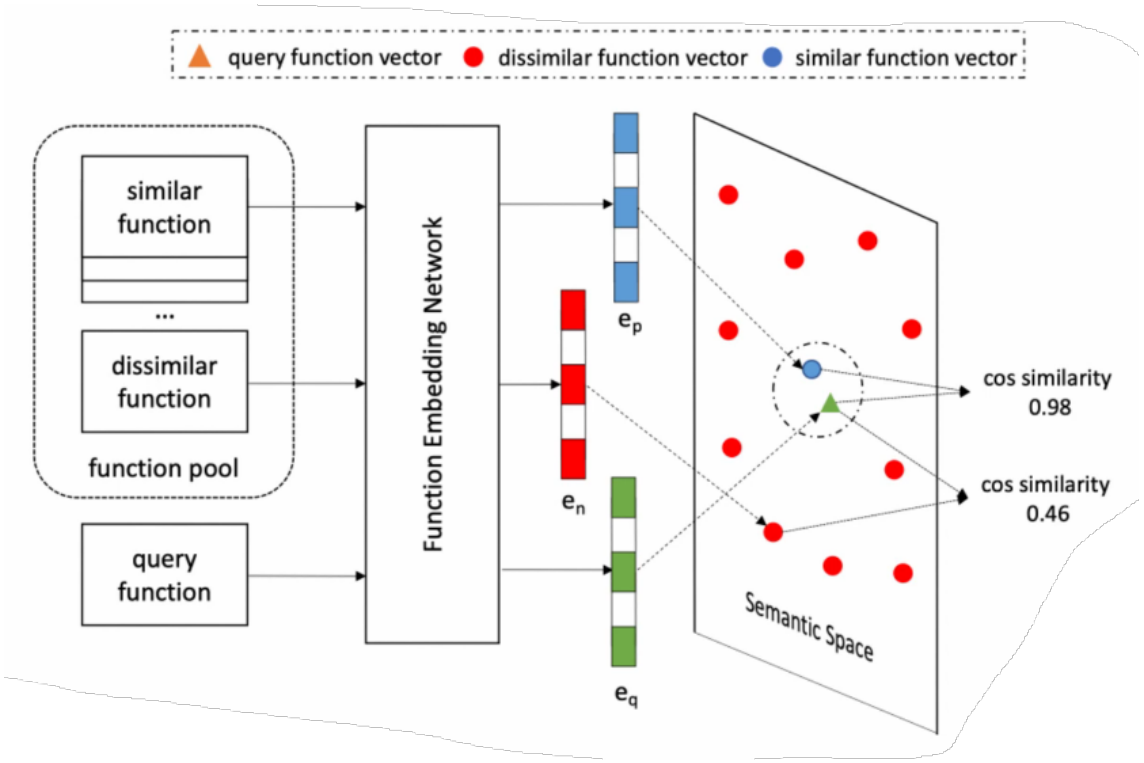
- 嵌入
  - 在模式识别框架中属于特征提取
  - 在保留图特征的前提下将图经过非线性变换成向量
  - 计算图之间的距离->计算数值向量之间的距离
- 以图嵌入为例：
  - structure2vec
  - 嵌套过程按照拓扑关系进行
  - 节点特征在嵌套过程中不断传播

非线性变换，常使用神经网络实现

更新后的嵌入结果

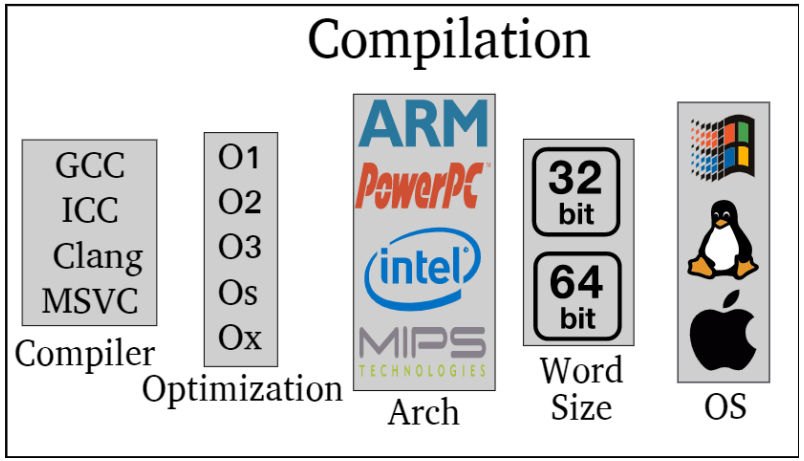
$$\mu_v^{t+1} = \mathcal{F}\left(x_v, \sum_{\mu \in \mathcal{N}(v)} \mu_v^t\right), \forall v \in \mathcal{V}$$

节点特征　　该节点所有邻居节点的嵌入结果

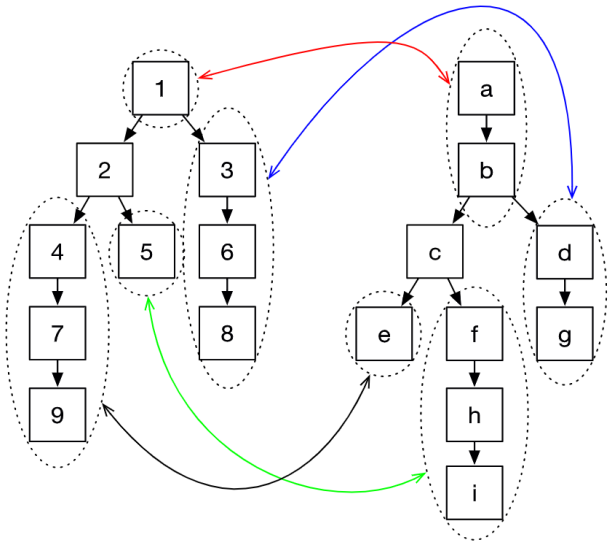


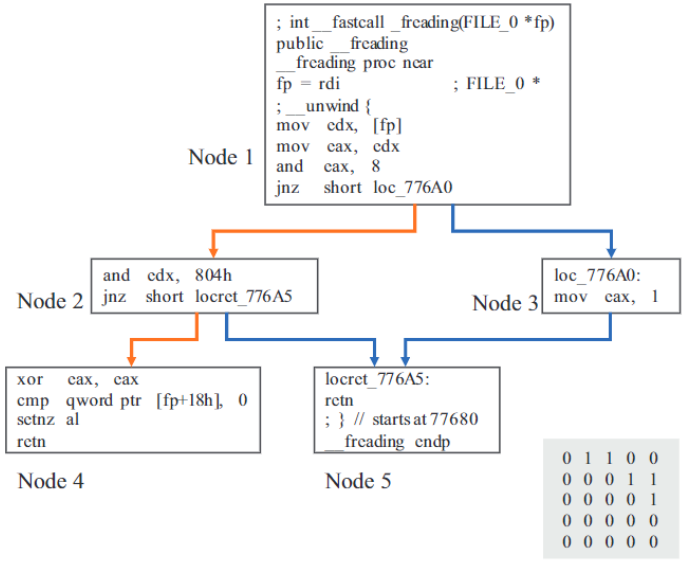
核心问题

- 不同编译过程带来控制流图差异
  - 指令集架构：X86，ARM，MIPS，……
  - 编译器：GCC，Clang，LLVM，……
  - 优化选项：O1，O2，O3，……
  - ……

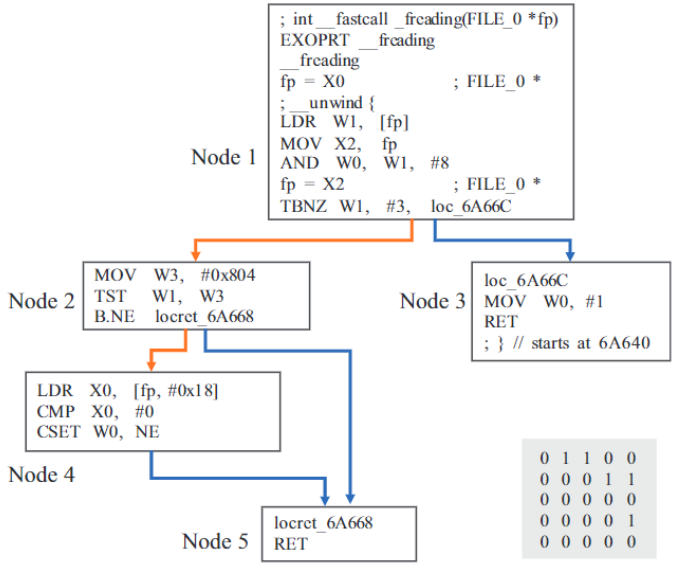


问题驱动创新：是否存在一种“不变”特征？





X86-64架构



ARM架构

T	目标	计算二进制代码片段间的相似性
I	输入	被查询的二进制代码片段；待搜索二进制代码库
P	处理	1. 利用word2vec提取基本块语义特征，完成单架构下指令嵌入 2. 利用孪生网络生成代码片段语义嵌入 3. 最大公共子序列算法计算相似度
O	输出	二进制代码片段相似度值

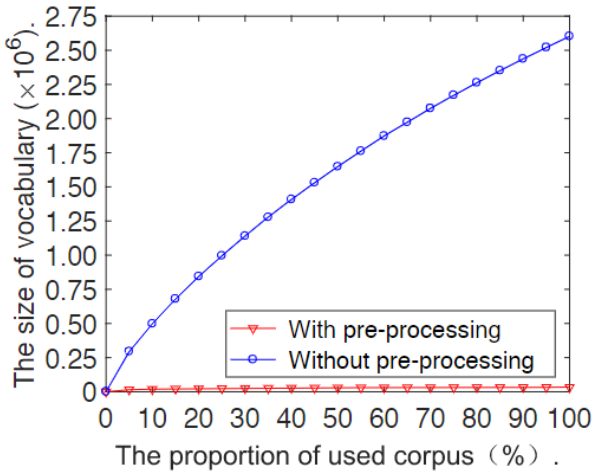
P	问题	提取二进制程序语义信息用于相似性分析
C	条件	二进制程序可被正常反汇编且未被混淆
D	难点	1. 如何提取二进制程序的语义信息 2. 如何实现跨架构的语义对比
L	水平	NDSS2019（CCF-A）

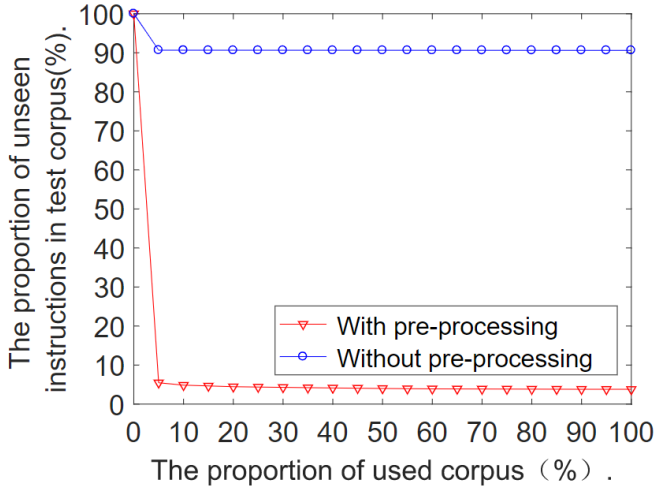
InnerEye 指令嵌入

- 词 ⇔ 汇编指令
  - mov edx，200
  - 词数量过于庞大?
- 句子 ⇔ 基本块
  - 句子：由词组成的序列
  - 基本块：由汇编指令组成的序列
- 预处理
  - 数值常量替换为0，保留正负号
  - 字符串替换为 <STR>
  - 函数名称替换为 FOO
  - 其他符号常量替换为 <TAG>

```
MOVL %ESI, $.L.STR.31
MOVL %EDX, $3
MOVQ %RDI, %RAX
CALLQ STRNCMP
TESTL %EAX, %EAX
JE .LBB0_5
```

```
MOVL ESI, <STR>
MOVL EDX, 0
MOVQ RDI, RAX
CALLQ FOO
TESTL EAX, EAX
JE <TAG>
```





InnerEye 指令嵌入

- word2vec
  - 将“词”映射至向量空间
  - 上下文相近的词在向量空间中距离更近
- 核心问题
  - 如何实现跨架构语义比较?

她们 夸 吴彦祖 帅 到 没朋友
她们 夸 阿伟 帅 到 没朋友

阿伟 = 吴彦祖

They say I'm too handsome to have any friends

$$J(\omega) = \frac{1}{T}\sum_{t=1}^{T}\sum_{\omega_k \in C_t}\left[\log P(\omega_k|\omega_t)\right]$$

句子长度

依次分析句字中所有词

在当前上下文条件下出现该词的概率

Source code

```
numblocks = (tmp_len+blocksize-1)/blocksize;
if(numblocks > pre_comp->numblocks)
```

X86-64 assembly

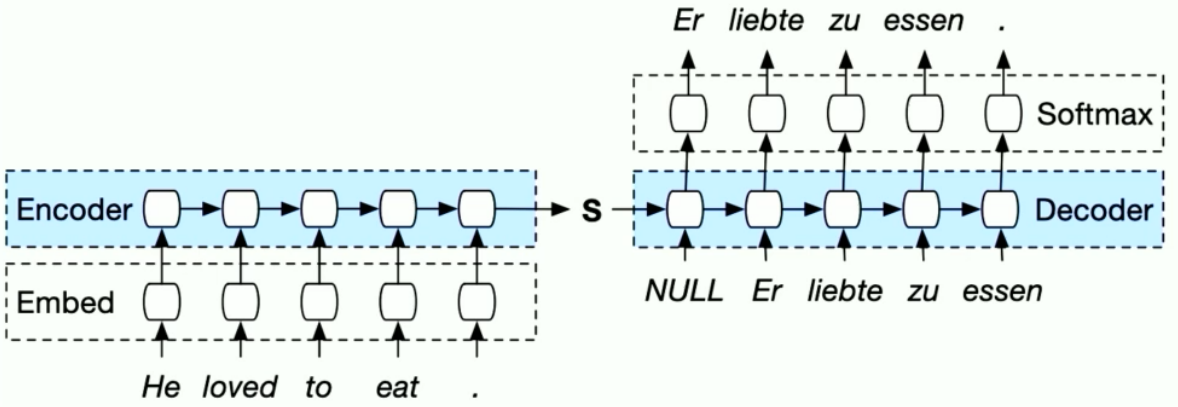
```
movq %rsi,80(%rsp)
addq %rax,%rsi
addq %rax,$-1
xorl %edx,%edx
divq %rsi
movq %rdx,96(%rsp)
cmpq %rax,16(%rdx)
jbe  .LBB2_68
```

ARM assembly

```
adds r1, r2, r1
adc  r7, r3, r0
subs r0, r1, #1
sbc  r1, r7, #0
bl   __udivdi3
ldr  r3, [sp, #60]
ldr  r2, [r3, #16]
ldr  r3, [r3, #20]
subs r2, r2, r0
sbcs r2, r3, r1
bhs  .LBB2_120
```

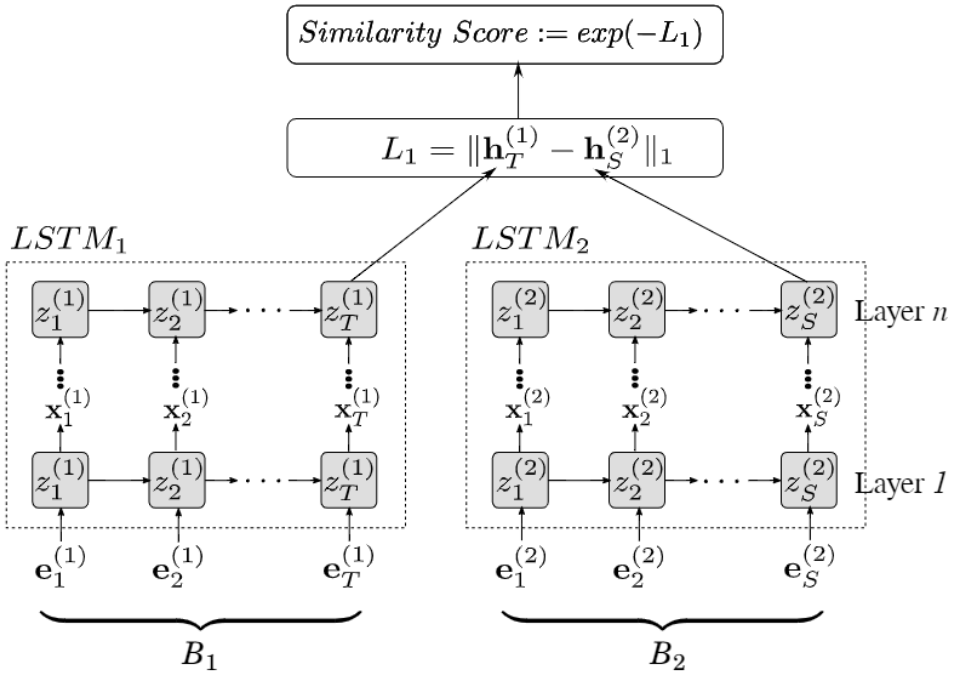
InnerEye 代码片段嵌入

- 文本翻译
  - 利用编码器计算语义编码
  - 根据语义编码恢复出另一种语言中的文本内容
- 代码片段嵌入
  - 利用LSTM计算语义编码
  - 孪生网络架构进行训练



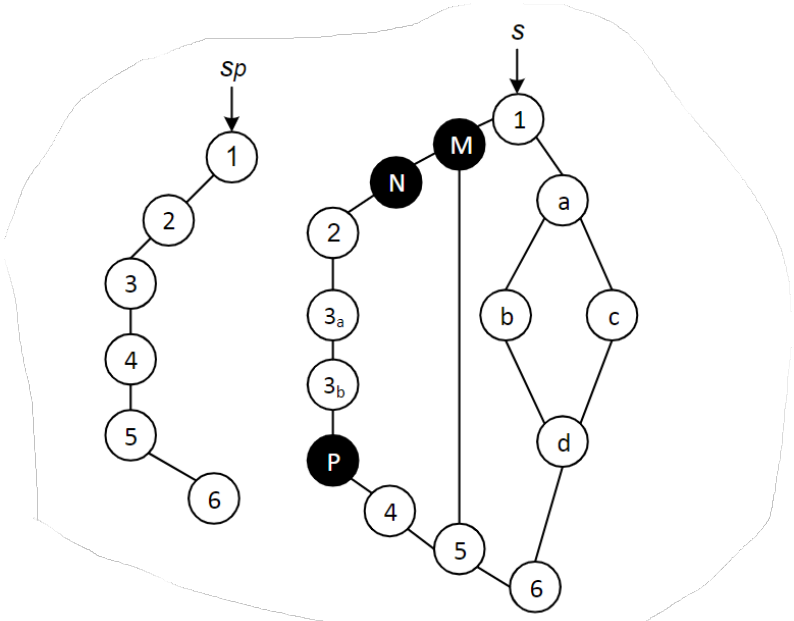
标签（相似/不相似）　　LSTM输出值的差异

$$\min \sum_{i=1}^{N} (y_i - \exp(-\|h_t - h_s\|))$$



InnerEye 函数相似度计算

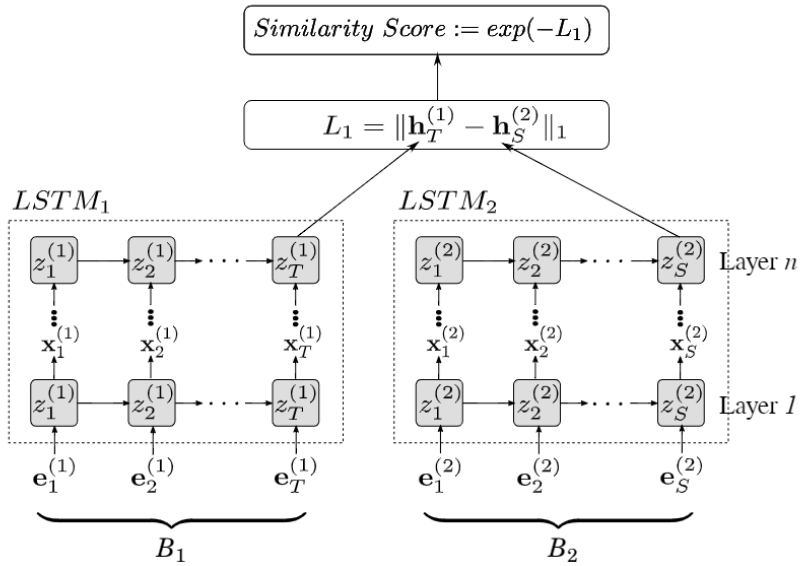
- 控制流图切片
  - 针对用于检索的函数，提取一条代表性片段
  - 针对目标函数，提取所有独立代码片段
- 同架构相似性判别
  - 计算最大公共子序列（Longest Common Subsequence，LCS）
- 跨架构相似性判别
  - LSTM+孪生网络



函数间最大公共子序列的长度

$$\psi(P,T) = \frac{\max|LCS(P,P_i^t)|}{|P|}$$

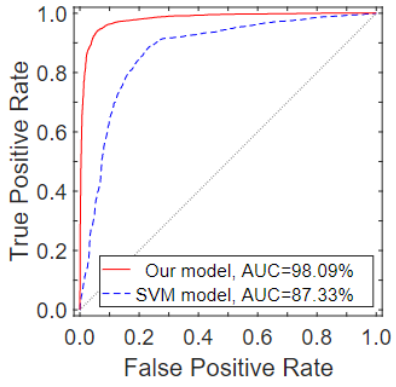
“代表性片段”的长度



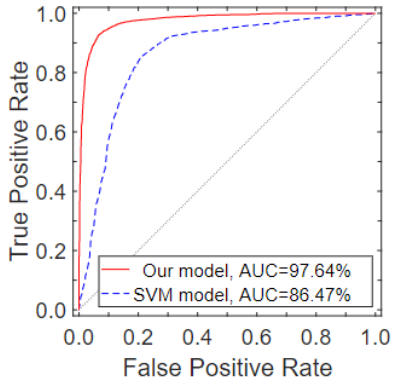
InnerEye 实验&总结

- 语义特征显著提升算法性能
- 支持跨优化选项/架构分析
- 后续发展
  - 领域内大量迁移/融合自然语言处理领域技术
  - 语义特征的提取和分析成为主流研究方向

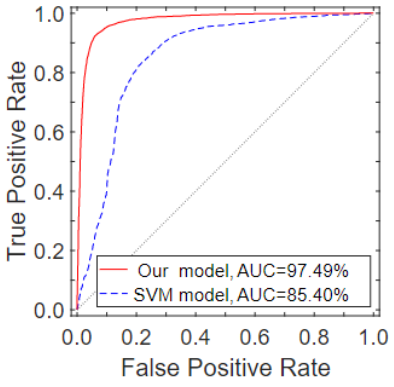
- 控制流图结构特征退出历史舞台?



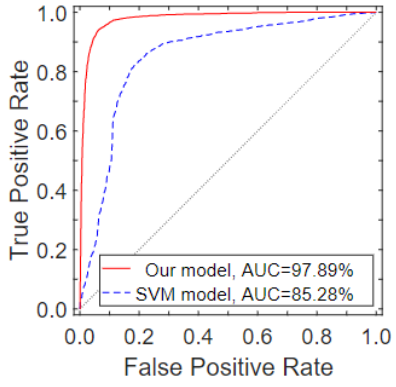
(a) O1



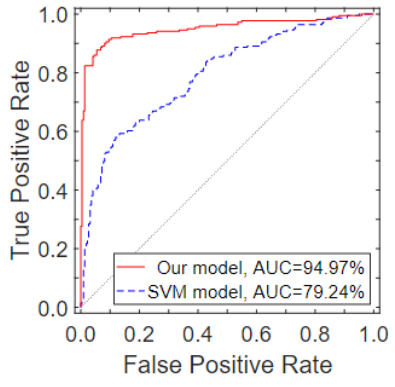
(b) O2



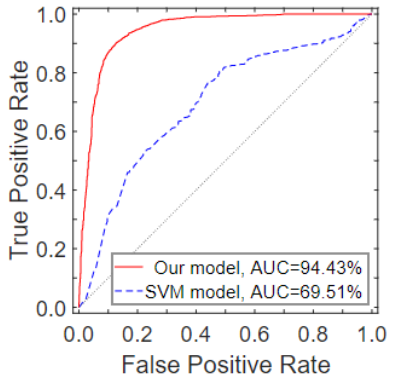
(c) O3



(d) Cross-opt-levels



(e) Large basic blocks in O3



(f) Small basic blocks in O3

x86	*ARM*	*x86*	*ARM*	*x86*	*ARM*

IMULQ R13,RAX,0
XORL EDX,EDX
MOVQ RBP,[RSP+0]
DIVQ RBP
JMP <TAG>
	MOV R1,R0
	LDR R6,[SP+0]
	CMP R0, 0
	BEQ <TAG>
		XORL R14D,R14D
		TESTQ RBP,RBP
		JE <TAG>
			LDMIB R5,R0,R1
			CMP R0,R1
			BHS <TAG>
				MOVL [RSP+0],R14D
				MOVQ RAX,[RSP+0]
				CMPB [RAX],0
				MOVQ [RSP+0],R13
				MOVQ [RSP+0],R15
				JNE <TAG>
					SUB R2,R1,0
					MOV R10,0
					CMP R2,0
					MOV R9,0
					BHI <TAG>

T	目标	分析非语义特征在二进制相似性分析中的作用
I	输入	被查询的二进制代码片段；待搜索二进制代码库
P	处理	1. 统计特征提取 2. 特征选择（贪心算法） 3. 相似度值计算
O	输出	二进制代码片段相似度值

P	问题	在不使用语义特征的前提下提升相似性分析准确率
C	条件	二进制程序可被正常反汇编且未被混淆
D	难点	不同限定条件下的特征选择
L	水平	2022 SCI 1区期刊

- 统计特征提取（共47维）
  - 基本块、边、循环数量
  - 各类指令个数
  - 节点入度、出度
  - ……
- 相似度计算
  - 特征间差异

Category	Features	Count
CFG	# of basic blocks, edges, loops, SCCs, and back edges # of all, arith, data transfer, cmp, and logic instrs. # of shift, bit-manipulating, float, misc instrs. # of arith + shift, and data transfer + misc instrs. # of all/unconditional/conditional control transfer instrs. Avg. # of edges per a basic block Avg./Sum of basic block, loop, and SCC sizes Avg. # of all, arith, data transfer, cmp, and logic instrs. Avg. # of shift, bit-manipulating, float, misc instrs. Avg. # of arith + shift, and data transfer + misc instrs. Avg. # of all/unconditional/conditional control transfer instrs.	41
CG	# of callers, callees, imported callees # of incoming/outgoing/imported calls	6
	Total	47

$$\delta(A_f, B_f) = \frac{|A_f - B_f|}{|\max(A_f, B_f)|}$$

特征数值差

特征中较大的值（用于标准化至0-1之间）

  - 二进制相似度

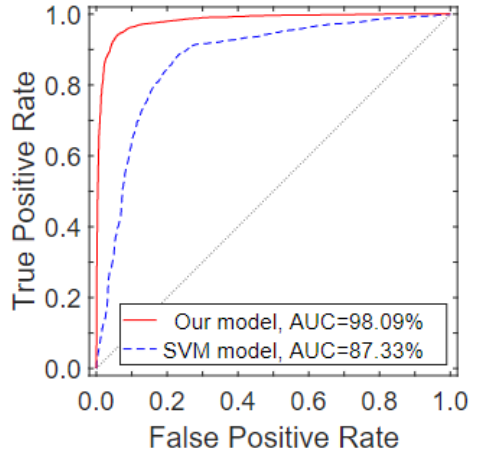
$$s(A, B) = 1 - \frac{\delta(A_{f1}, B_{f1}) + \cdots + \delta(A_{f2}, B_{f2})}{N}$$

特征间差异

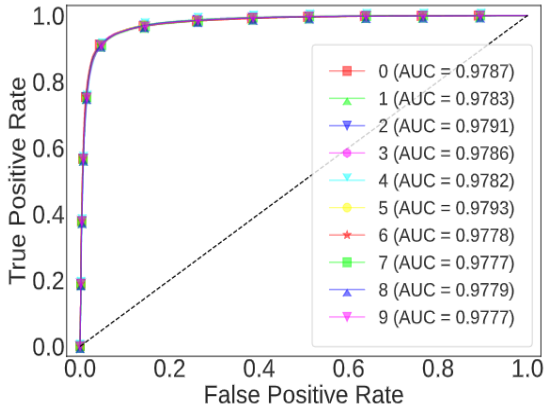
使用的特征总数

- 特征选择
  - 贪心算法
    - 从候选特征中不断选择提升AUC值最有效的特征
    - AUC值不再提升时停止选择
- 证明结构特征在二进制相似性分析中的价值
- 局限性
  - 针对不同对比情况选取不同特征，不具有泛化性
  - 存在“过拟合”

		Opt Level			Compiler				Arch					
	Rand.	O0 vs. O3	O2 vs. O3	Rand.	GCC 4 vs. GCC 8	Clang 4 vs. Clang 7	GCC vs. Clang	Rand.	x86 vs. ARM	x86 vs. MIPS	ARM vs. MIPS	32 vs. 64	LE vs. BE	Rand.
Avg. # of Selected Features	4.0	4.0	6.8	5.3	7.1	10.5	6.8	3.0	12.0	6.4	7.1	9.4	7.7	9.0
Avg. TP-TN Gap of Grey	0.53	0.52	0.56	0.53	0.58	0.59	0.53	0.56	0.54	0.54	0.54	0.56	0.56	0.55
ROC AUC	0.99	0.99	1.00	0.99	1.00	1.00	0.99	0.99	1.00	0.99	1.00	1.00	1.00	1.00
Average Precision (AP)	0.99	0.99	1.00	0.99	1.00	1.00	0.99	0.99	1.00	0.99	1.00	1.00	1.00	1.00



(a) O1



T	目标	计算二进制代码片段间的相似性
I	输入	被查询的二进制代码片段；待搜索二进制代码库
P	处理	1. 提取二进制反汇编图（Binary Disassembly Graph） 2. 二进制反汇编图中的汇编指令嵌入为向量 3. 利用GCN实现函数嵌入
O	输出	二进制代码片段相似度值

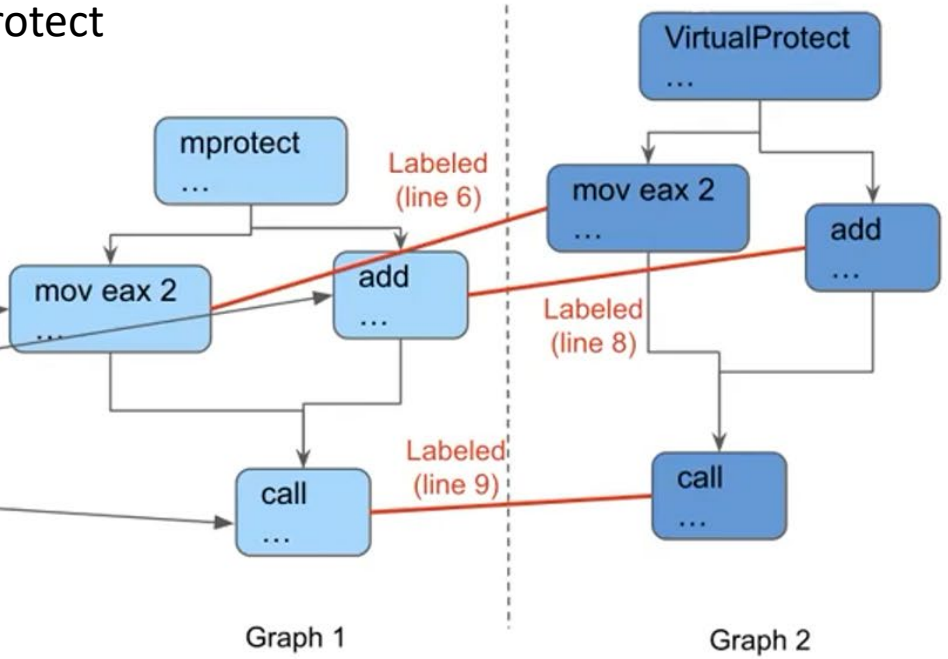
P	问题	现有基于语义特征的方法性能较差
C	条件	二进制程序可被正常反汇编且未被混淆
D	难点	在保留语义特征的前提下增加结构特征
L	水平	ISSTA2022（CCF-A）

- 为什么现有方法没有起到应有的效果?
  - 忽视二进制结构特征
  - 源代码中包含操作系统下的特有代码
  - 在数据标注时无法顾及特有代码匹配
  - 未标注基本块间的语义相似度无法学习

Windows编译时使用mprotect

Linux编译时使用VirtualProtect

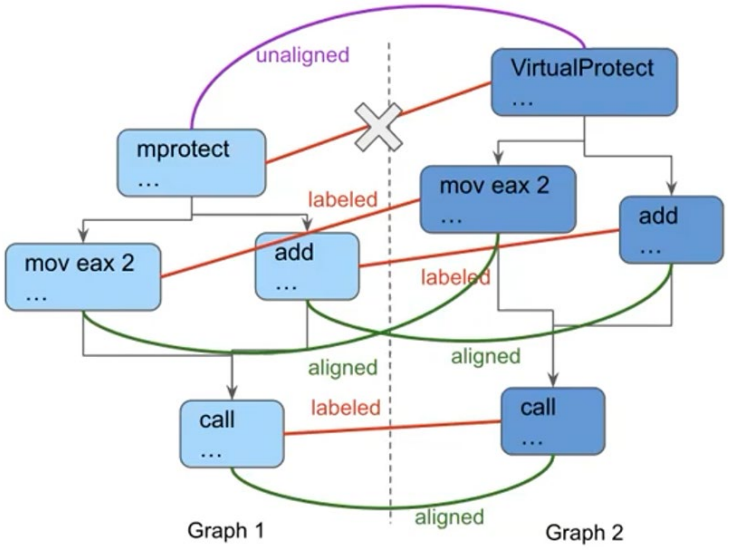
```
1.  #if !defined(_WIN32)
2.      if (mprotect(sh.map_result, pgsize, PROT_NONE) < 0)
3.  #else
4.      if (VirtualProtect(sh.map_result, pgsize, PAGE_NOACCESS,
                           &flOldProtect) == FALSE)
5.  #endif
6.          ret = 2;
7.      else
8.          ret += 2;
9.  printf("Hello world!")
```

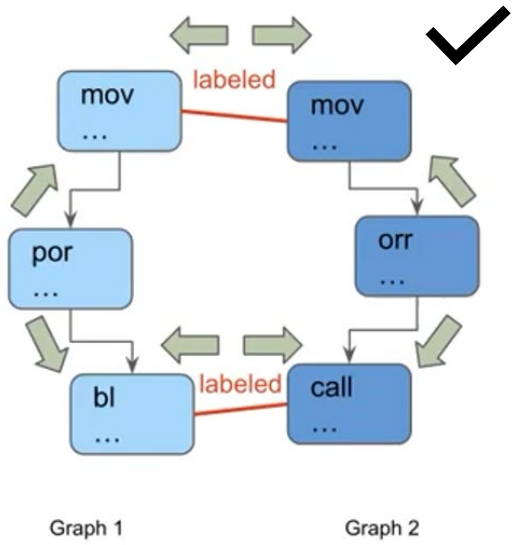


XBA 问题分析

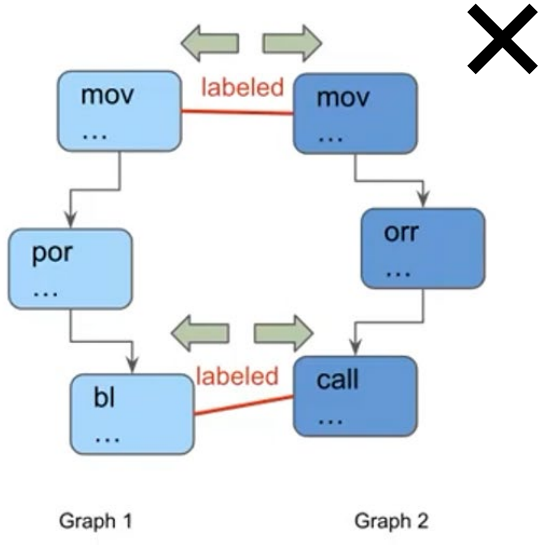
- 未标注基本块占比3% - 40%
- 解决方案
  - 额外学习基本块之间的结构特征 ⇔ 二进制反汇编图+图表示学习
  - 学习未标注基本块间的语义相似度 ⇔ 半监督训练

	Configurations Platform(s) / Opt. Level(s)	# of Basic Blocks 1-to-1 Aligned	Unaligned
libc.so	x86_64 ↔ ARMv7 / O1	29,626	**11,531**
	x86_64 ↔ ARMv7 / O2	23,274	**14,501**
	x86_64 ↔ ARMv7 / O3	21,106	**14,307**
	x86_64 / O1 ↔ O2	31,880	**6,315**
	x86_64 / O1 ↔ O3	29,166	**7,067**
	x86_64 / O2 ↔ O3	33,260	**1,910**
libcrypto.so	x86_64 ↔ ARMv7 / O0	133,282	**4,621**
	x86_64 ↔ ARMv7 / O1	79,580	**16,836**
	x86_64 ↔ ARMv7 / O2	67,780	**22,595**
	x86_64 ↔ ARMv7 / O3	62,404	**21,837**





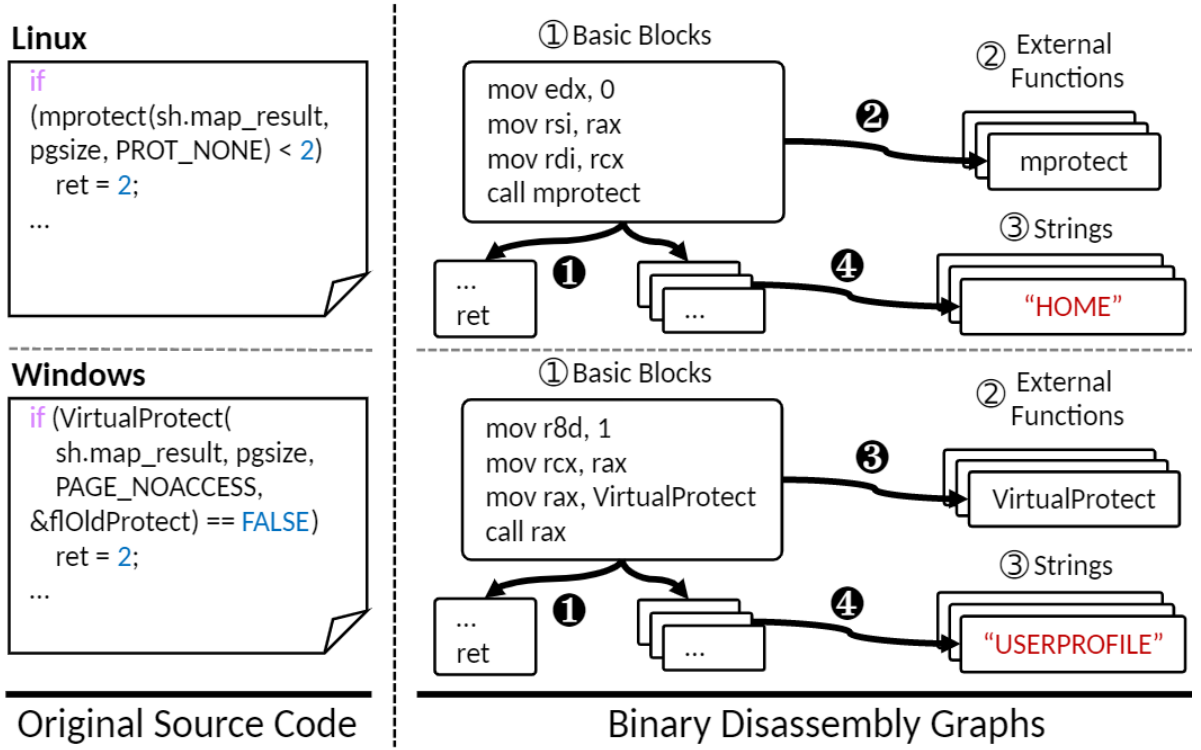
XBA



InnerEye

XBA 二进制反汇编图

- 节点（实际应用中均为嵌入向量）
  - ① 基本块：指令序列
  - ② 外部函数：函数名
  - ③ 字符串
- 边
  - ① 跳转关系（同控制流图）
  - ② 直接调用
  - ③ 间接调用
  - ④ 基本块使用了某字符串
- 与InnerEye对比
  - 额外保留字符串信息、外部函数调用信息



Linux: read =? .sqliterc

Windows: ReadFile

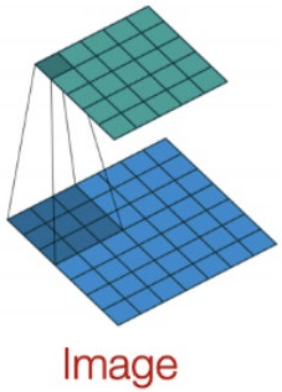
- 图表示学习
  - 将节点特征与结构信息同时输入，由算法自行决定利用哪些信息
- GCN
  - 利用周围节点的信息来推导节点自身特征
  - 在此基础上额外考虑自身信息
  - 标准化处理
- GCN仅需少量标注数据即可实现较高分类性能

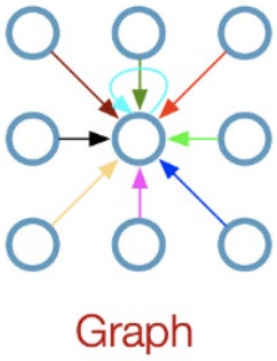
后一层特征　邻接矩阵　前一层特征　权重

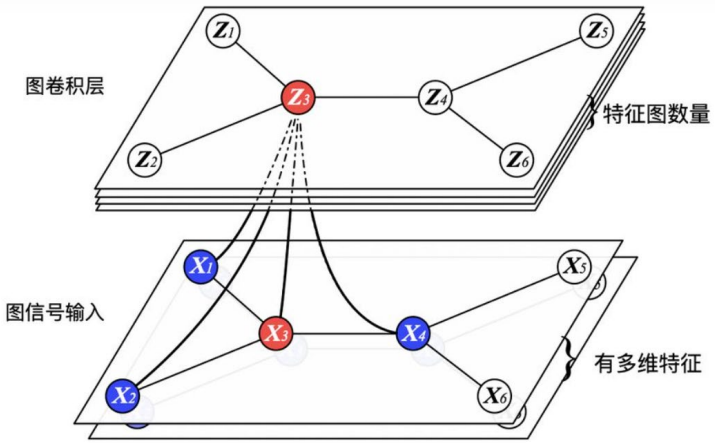
$$H^{l+1} = \sigma(AH^lW^l)$$

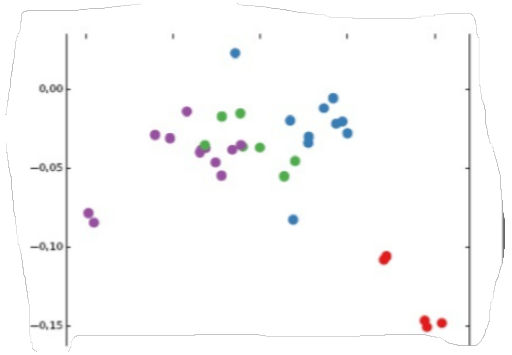
$$\hat{A} = A + I$$

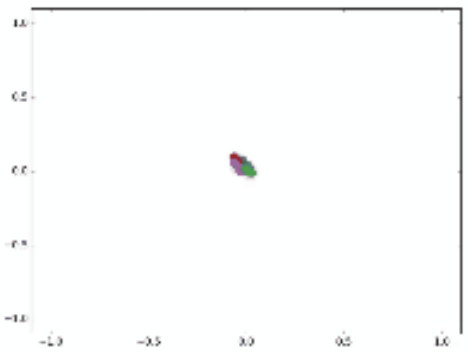
$$H^{l+1} = \sigma(D^{\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}H^lW^l)$$



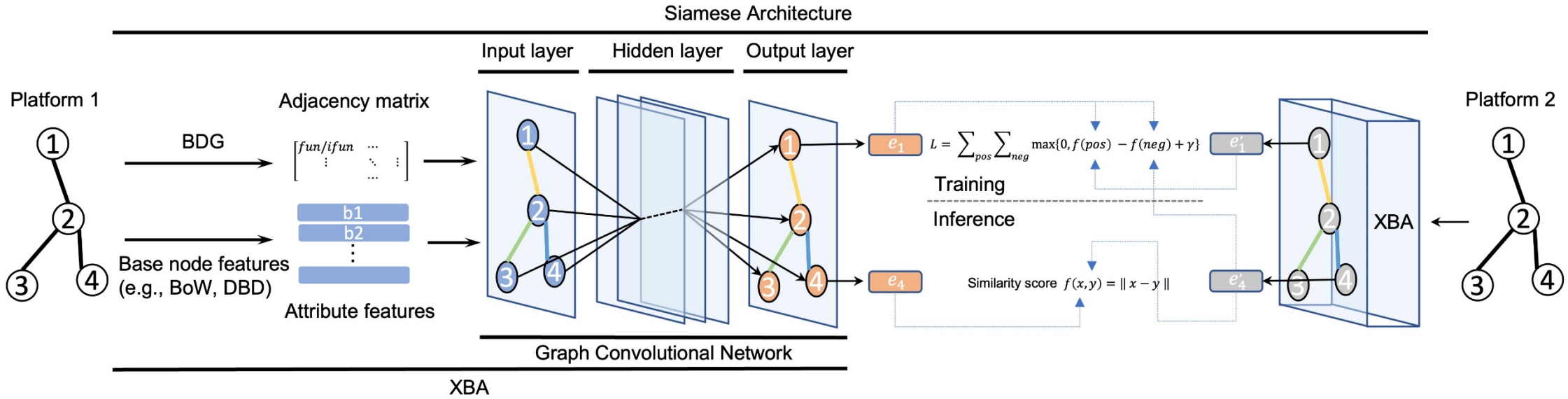








- XBA同样使用孪生网络训练模型
- 可扩展性
  - XBA方法只涉及图构建及图表示学习
  - 指令嵌入可使用其他先进方法



XBA 实验结果

Cross-OS Configuration	Dataset	Linux→Windows Hits@1	Linux→Windows Hits@10	Linux→Windows Hits@50	Linux→Windows Hits@100	Windows→Linux Hits@1	Windows→Linux Hits@10	Windows→Linux Hits@50	Windows→Linux Hits@100
InnerEye	SQLITE3	70.38	79.27	85.90	90.10	71.52	80.06	86.78	89.76
BoW Encoding	SQLITE3	84.13	89.60	92.83	96.94	85.80	91.59	94.10	94.57
BoW Encoding + **XBA**	SQLITE3	97	99.32	99.65	99.71	97.01	99.41	99.75	99.82
DeepBinDiff	SQLITE3	78.14	86.14	90.07	91.45	80.13	87.32	90.80	92.35
DeepBinDiff + **XBA**	SQLITE3	94	98.14	99.31	99.54	94.57	98.64	99.32	99.54
InnerEye	OPENSSL	59.10	70.86	81.04	86.20	63.19	73.80	82.48	85.17
BoW Encoding	OPENSSL	70.47	80.80	87.96	89.28	78.59	85.08	90.66	92.38
BoW Encoding + **XBA**	OPENSSL	88.17	99.05	99.65	99.79	90.33	99.10	99.69	99.82
DeepBinDiff	OPENSSL	63.48	79.12	86	88.50	84.65	94.66	96.62	97.14
DeepBinDiff + **XBA**	OPENSSL	88.23	97.81	99.22	99.43	90.24	98.46	99.39	99.62
InnerEye	CURL	73.56	81.06	90.20	92.87	74.87	83.15	89.47	91.47
BoW Encoding	CURL	85.21	90.99	97.17	97.76	87.63	92.15	94.34	98.33
BoW Encoding + **XBA**	CURL	88.59	97.83	99.17	99.35	88.08	96.82	98.22	98.53
DeepBinDiff	CURL	76.57	85.46	89.96	91.84	79.17	87.44	90.95	92.55
DeepBinDiff + **XBA**	CURL	88.27	98.08	99.39	99.62	89.05	97.75	98.68	98.97
InnerEye	HTTPD	72.27	83.98	95.09	96.27	73.43	84.38	91.57	92.99
BoW Encoding	HTTPD	84.02	90.79	97.52	98.24	86.61	91.08	93.60	97.89
BoW Encoding + **XBA**	HTTPD	87.95	98.43	99.44	99.64	87.46	97.65	98.88	99.24
DeepBinDiff	HTTPD	68.56	83.14	90.60	92.77	65.22	79.78	87.19	90.76
DeepBinDiff + **XBA**	HTTPD	81.57	95.89	98.19	98.92	81.82	95.22	97.59	98.48
InnerEye	LIBCRYPTO	68.96	73.50	78.26	80.42	68.15	73.01	77.11	79.92
BoW Encoding	LIBCRYPTO	78.51	83.99	87.02	88.80	77.81	82.75	86.29	88.03
BoW Encoding + **XBA**	LIBCRYPTO	79.03	94.28	97.01	97.81	81.33	95.06	97.38	97.92
DeepBinDiff	LIBCRYPTO	68.47	79.78	83.71	85.32	67.79	78.80	82.47	84.25
DeepBinDiff + **XBA**	LIBCRYPTO	76.34	90.40	94.47	95.65	78.56	91.37	94.81	95.73

总结

- 应用总结
  - 同源漏洞判别
    - 搜索与已知漏洞相似的程序片段
  - 恶意软件检测
  - 补丁分析
    - 分析比较同一软件不同版本下的差异
  - 代码窃取检测
- 前沿发展
  - 无监督条件下的相似度分析
  - 语义关系扩展
    - 定位具有语义相关性的函数，如加密函数与解密函数
  - 抗混淆

参考文献

- KIM G, HONG S, FRANZ M, et al. Improving cross-platform binary analysis using representation learning via graph alignment[C]//Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis.2022:151-163.
- KIM D, KIM E, CHA S K, et al. 2022. Revisiting Binary Code Similarity Analysis using Interpretable Feature Engineering and Lessons Learned. IEEE Transactions on Software Engineering [J]: 1-23.
- ZUO F, LI X, YOUNG P, et al. Neural Machine Translation Inspired Binary Code Similarity Comparison beyond Function Pairs[C]//Network and Distributed Systems Security (NDSS) Symposium 2019.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！