Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

二进制程序开源成分分析

硕士研究生 邢继媛

2022年6月26日

内容提要

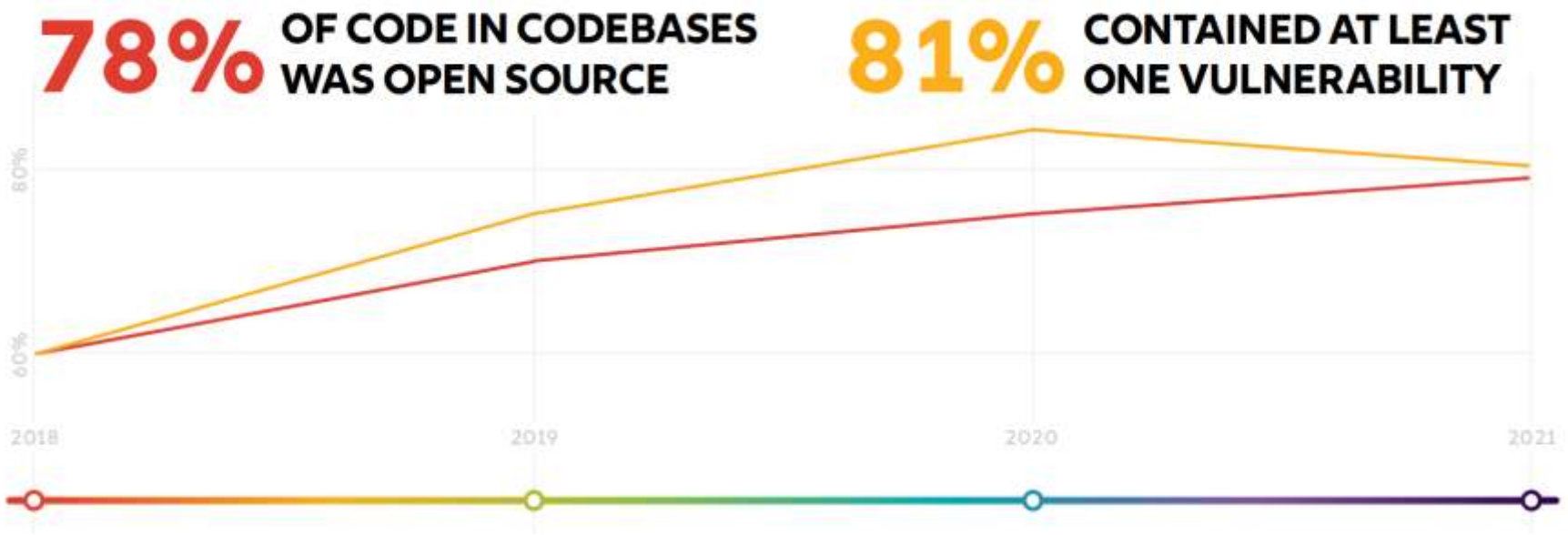
- 背景简介
- 基本概念
- 算法原理
- 应用总结
- 参考文献

预期收获

- 预期收获
  - 1. 了解二进制程序开源成分分析基本概念
  - 2. 掌握二进制代码/源代码匹配基本框架
  - 3. 掌握二进制程序开源成分分析基本原理

背景简介

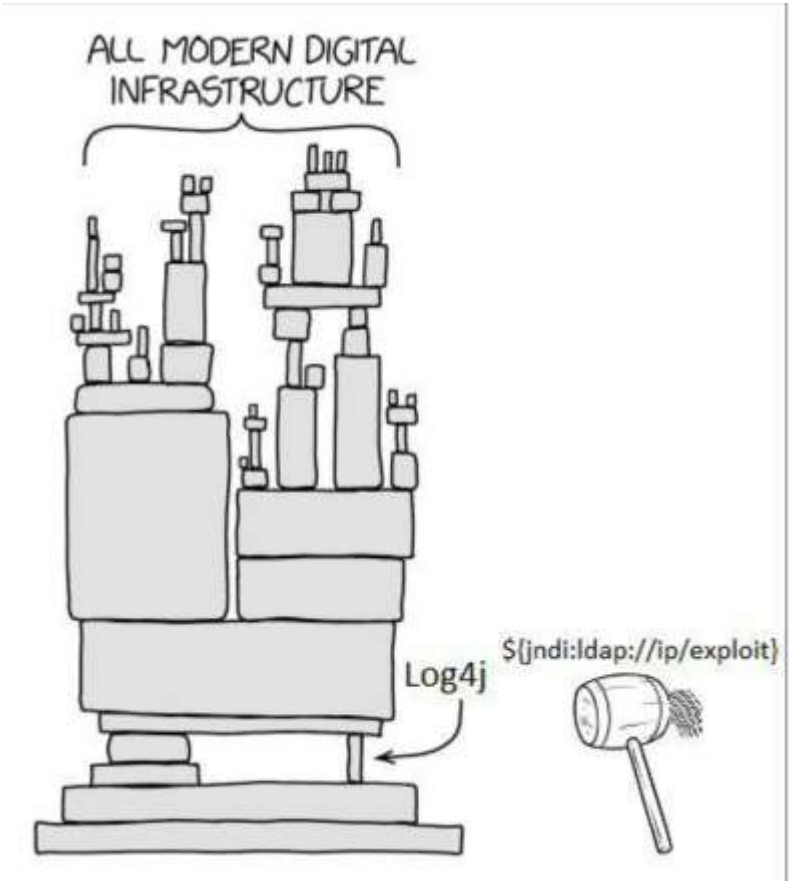
- 软件开发中第三方库的复用趋势逐渐增长
- 不受控制的复用将导致问题
  - 违反开源许可证、引入同源漏洞



2022开源安全和风险分析报告

背景简介

- Log4j2 “核弹级”漏洞（2021年12月）
  - Apache Log4j2（Log For Java）组件，基于 Java 的开源日志框架
  - 该组件非常受欢迎，被数百万企业应用程序和服务所使用
  - 漏洞后果
    - 设备远程受控
    - 敏感信息窃取
    - 设备服务中断



基本概念

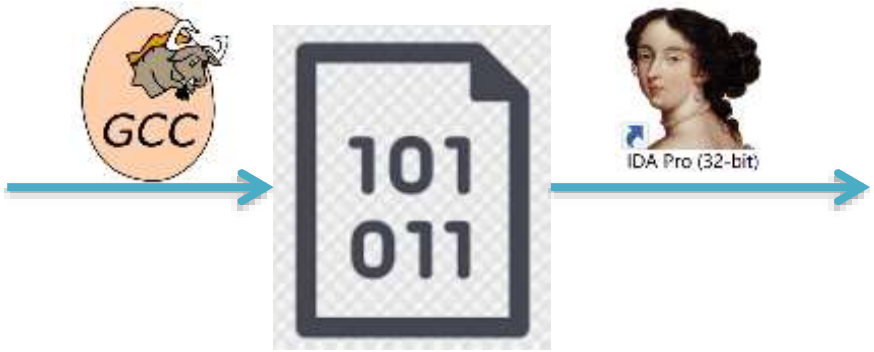
基本概念

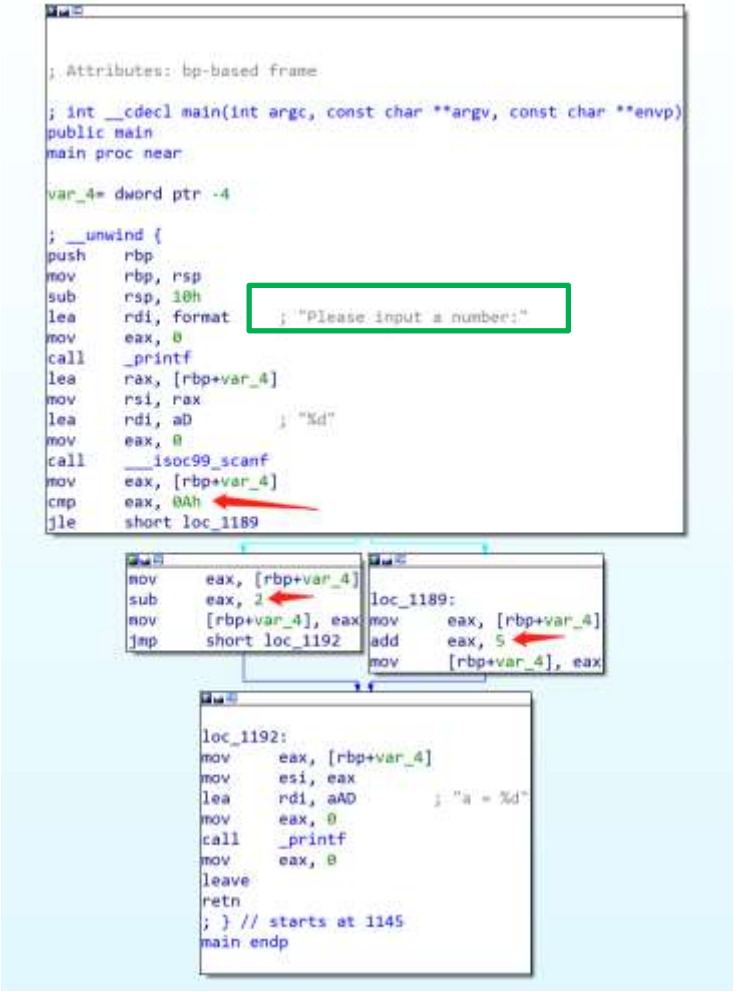
- TPL（Third-Party Library）
  - 可复用的软件组件，由原始开发供应商以外的其他各方开发
  - 可用于避免重复开发具有相同功能的软件，从而节省时间和资源
  - 缺乏来自第三方的支持，使用它会引入依赖性问题和安全问题
- TPL检测（TPL detection）
  - 软件成分分析 SCA（Software Composition Analysis）
  - 检测软件中第三方库使用情况

基本概念

- 反汇编
  - 源码->编译->二进制程序->反汇编-> 控制流图
- 控制流图（CFG）
  - 有向图 G（V，E）
    - 节点：基本块
    - 边：控制流

```c
int main()
{
    int a;
    printf("Please input a number:");
    scanf("%d", &a);
    if(a > 10){
        a = a - 2;
    }else{
        a = a + 5;
    }
    printf("a = %d", a);
    return 0;
}
```





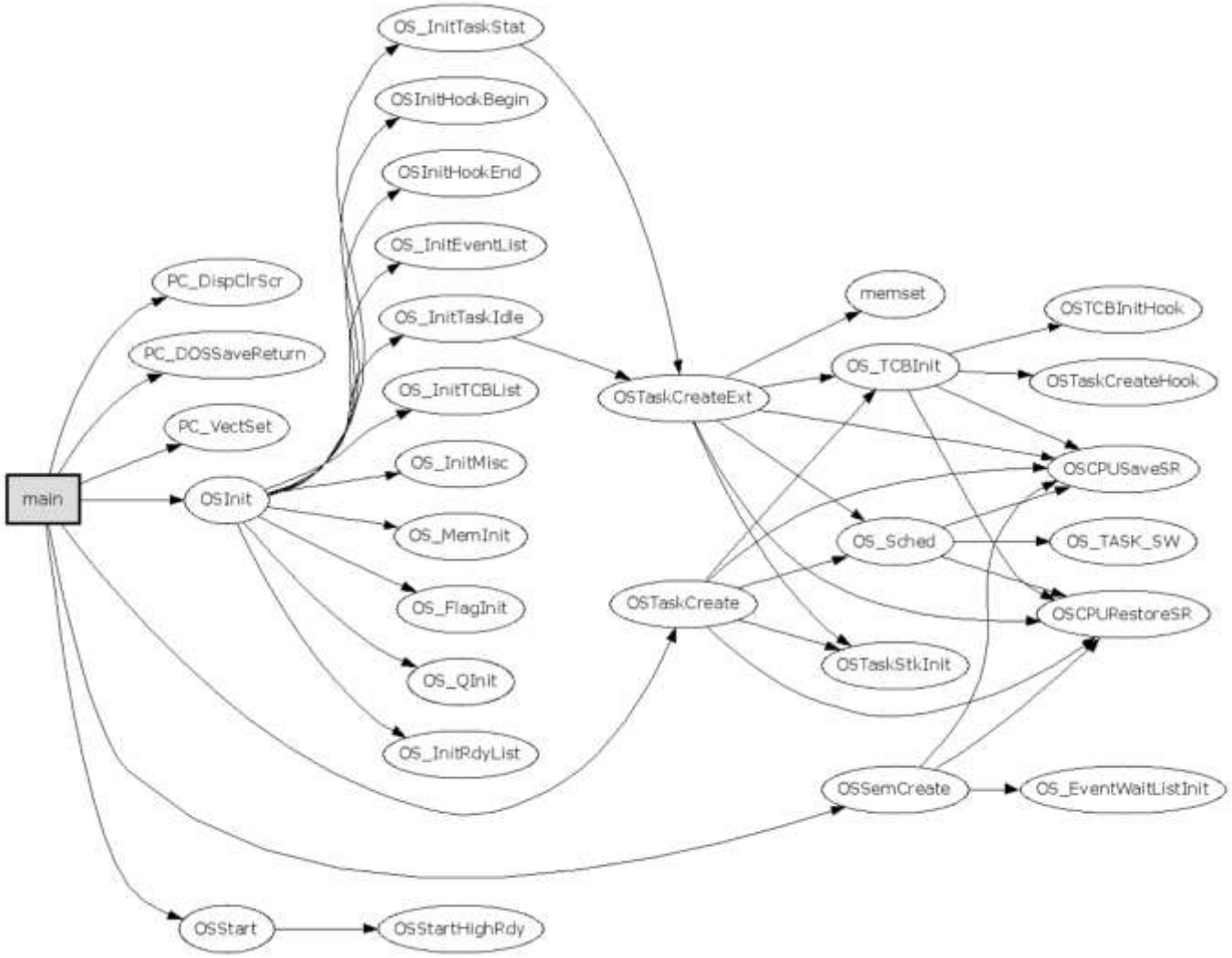
控制流图

基本概念

- 跨模态任务
  - 将一种类型的数据作为查询去检索另一种相关类型的数据
  - 主要的三种模态：自然语言（写/说等）、视觉信号（图片/视频等）、声音信号（声音编码/韵律等）

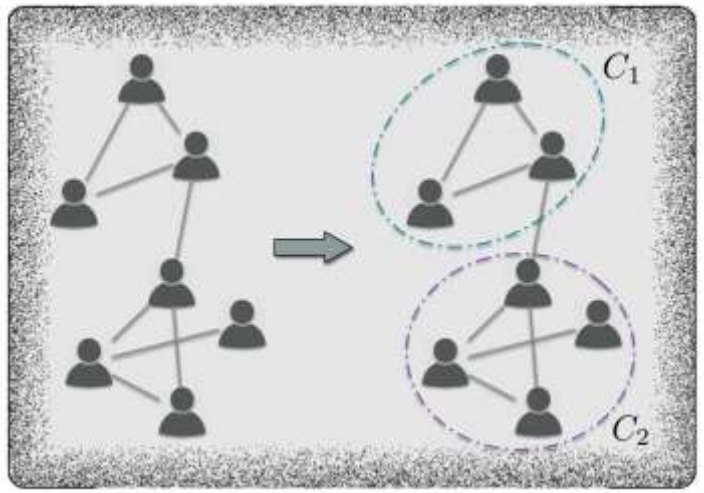
基本概念

- 函数调用图
  - 反应函数之间的调用关系
  - 节点：函数
  - 边：调用关系
  - 入度/出度



基本概念

- 社区检测任务
  - 在图结构中发现密集连接的子网络
  - 例：
    - Twitter/Facebook社交网络，具有共同兴趣或共同朋友的用户可能是同一个社区的成员
    - 在企业网络中，可以通过公司的内部关系将员工分组为社区
    - 在蛋白质网络中，社区检测有助于发现相似生物学功能的蛋白质



算法原理

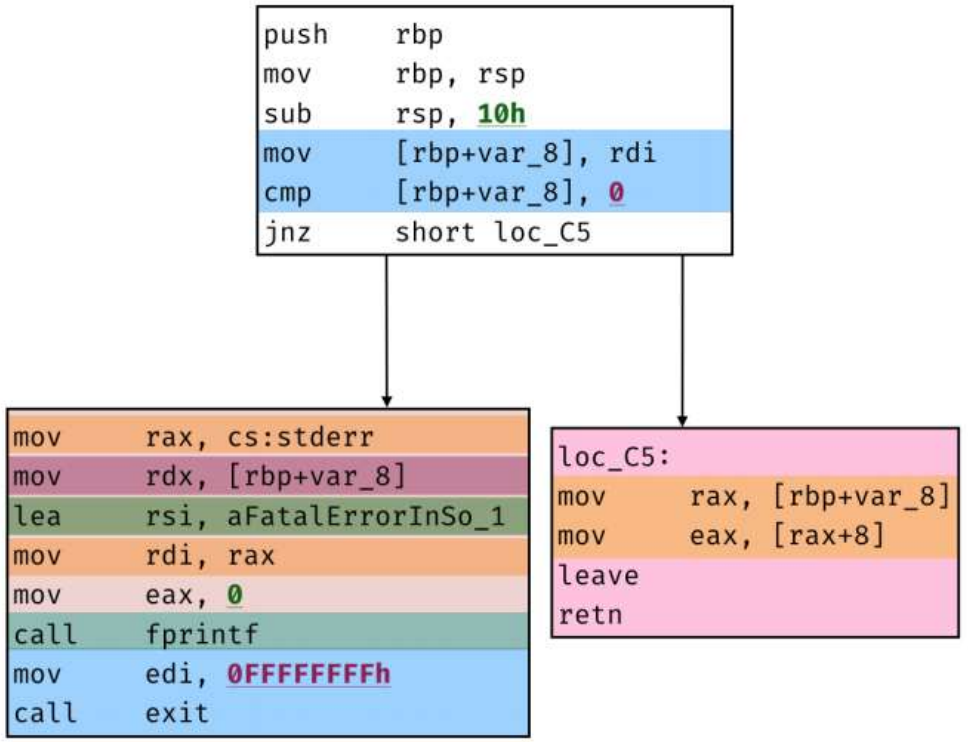
算法原理

T	二进制/源代码函数级匹配
I	二进制函数、源代码函数
P	1. 源代码/二进制函数语义特征提取 2. 立即数/字符串特征提取 3. 相似度计算
O	二分类

P	函数级别的源代码与二进制代码的相似特征非常少
C	二进制程序函数未混淆
D	提取源代码与二进制代码中隐藏的语义特征
L	2020 NeurIPS（人工智能顶会）

算法原理

- 函数源码对应的二进制函数的控制流图




```
.rodata:00000000000003B8 ; char aFatalErrorInSo_1[]
.rodata:00000000000003B8 aFatalErrorInSo_1 db 0Ah
.rodata:00000000000003B8                   db ' fatal error in SolveMap_nproc(%p)',0Ah
.rodata:00000000000003B8                   db ' bad input',0Ah,0
```

(a) Binary code of function "SolveMap_nproc"

```
int SolveMap_nproc (SolveMap *solvemap) {
  if (solvemap ==((void *)0)) {
    fprintf(stderr, "\n fatal error in SolveMap_nproc(%p)"
                    "\n bad input\n", solvemap) ;
    exit(-1) ;
  }
  return(solvemap→nproc) ;
}
```

(b) Source code of function "SolveMap_nproc"

Binary:	10h	0	0	0FFFFFFFFh
Source:		0		-1

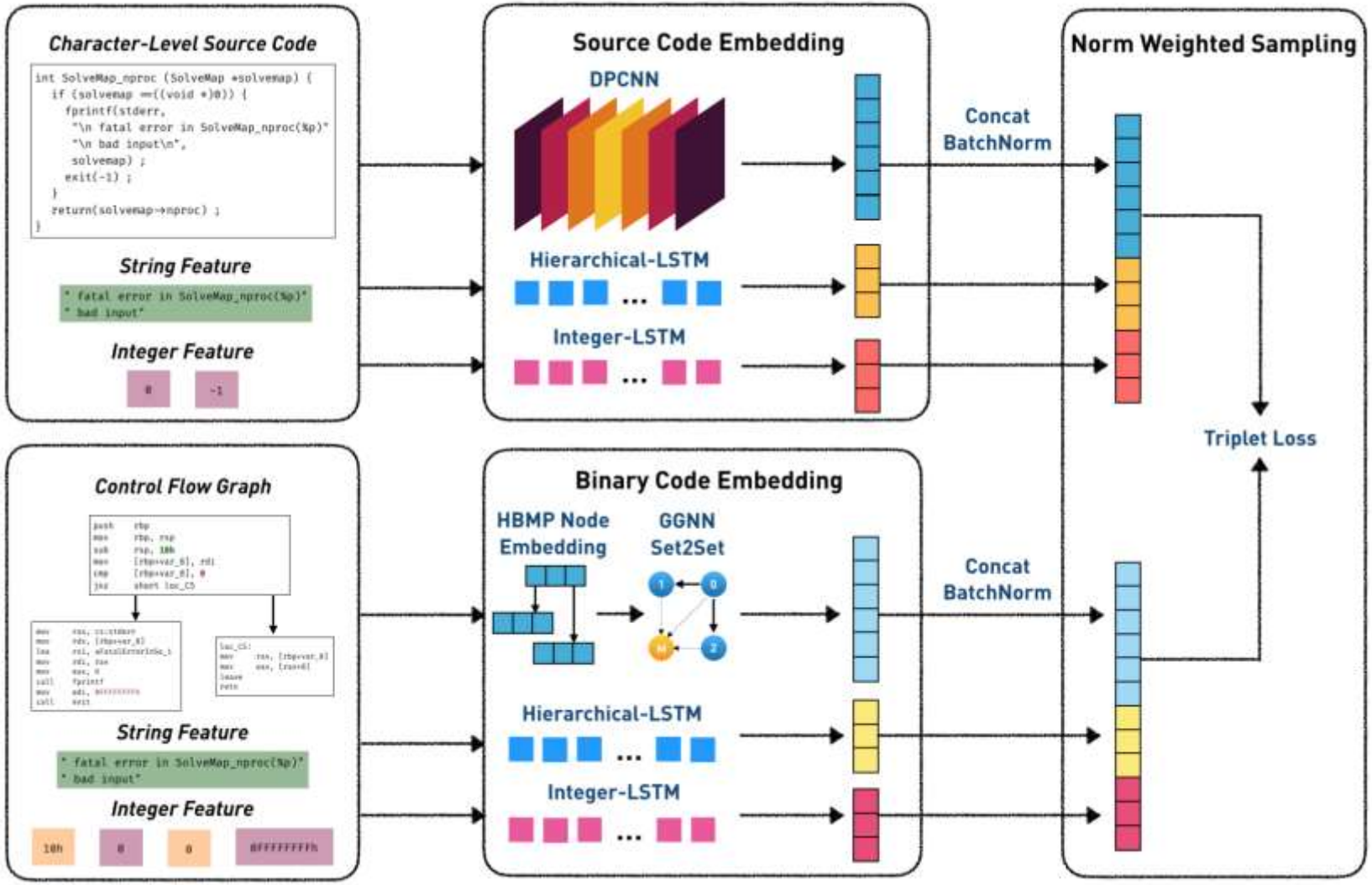
(c) Integer feature of function "SolveMap_nproc"

Binary:	fatal error in SolveMap_nproc(%p) bad input
Source:	fatal error in SolveMap_nproc(%p) bad input

(d) String feature of function "SolveMap_nproc"

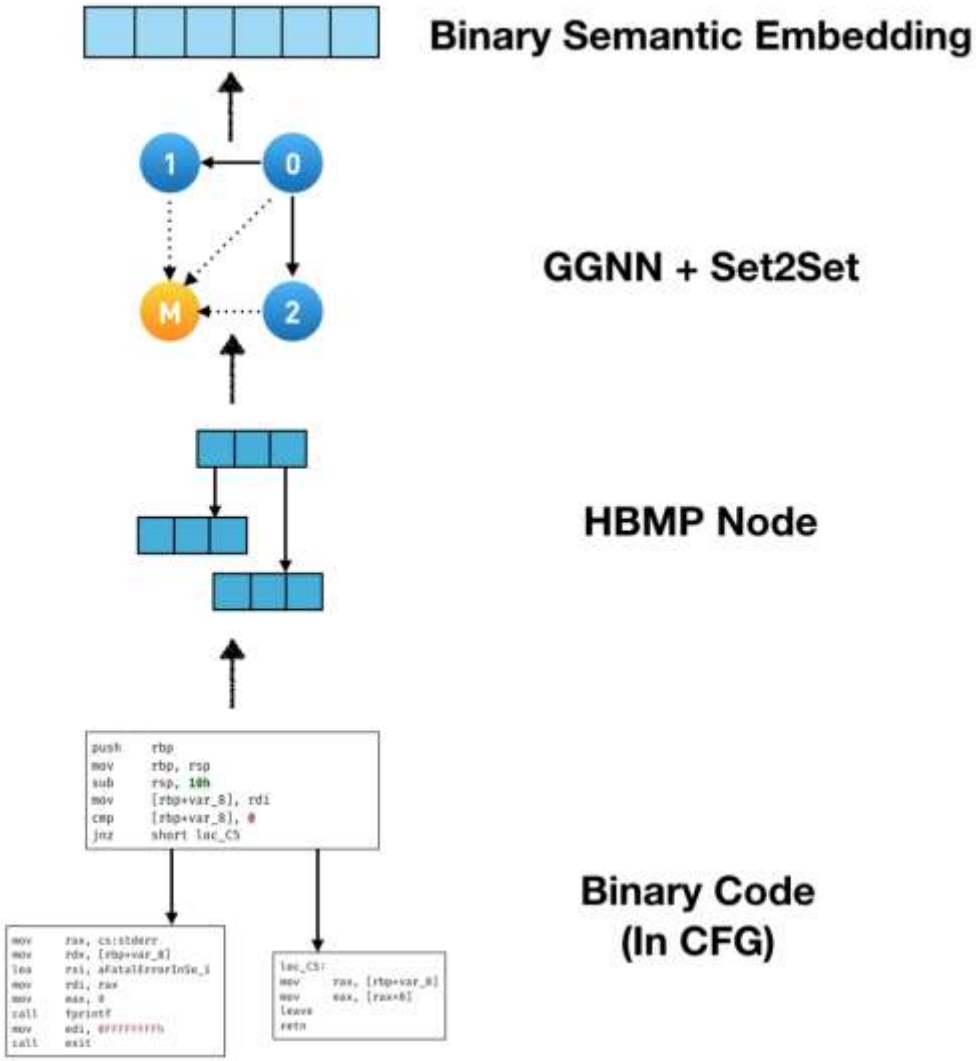
算法原理

- 跨模态框架


Character-Level Source Code
String Feature
Integer Feature
Source Code Embedding
DPCNN
Hierarchical-LSTM
Integer-LSTM
Concat
BatchNorm
Norm Weighted Sampling
Triplet Loss
Control Flow Graph
String Feature
Integer Feature
Binary Code Embedding
HBMP Node
Embedding
GGNN
Set2Set
Hierarchical-LSTM
Integer-LSTM
Concat
BatchNorm

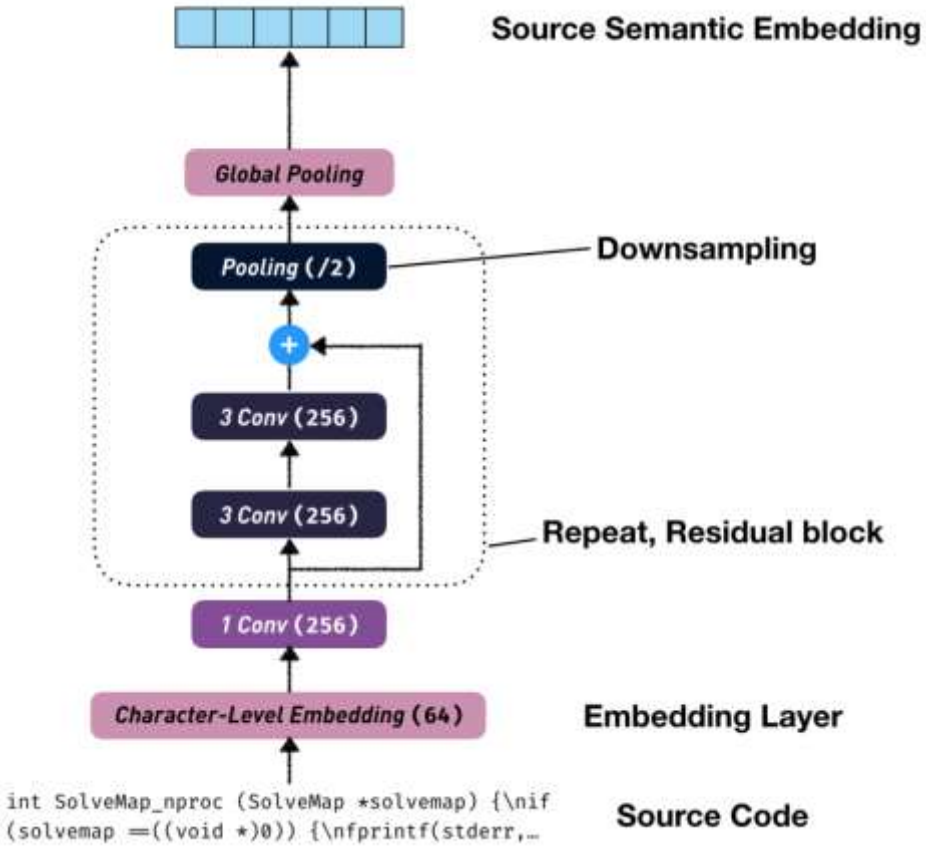
算法原理

- 二进制代码语义特征提取
  - HBMP
    - 句子嵌入算法
    - 生成节点的特征向量
  - GGNN
    - GNN+GRU
    - 结合图结构生成节点向量表示
  - Set2Set
    - 与输入顺序无关的图池化方法
    - 聚合节点特征生成函数向量表示



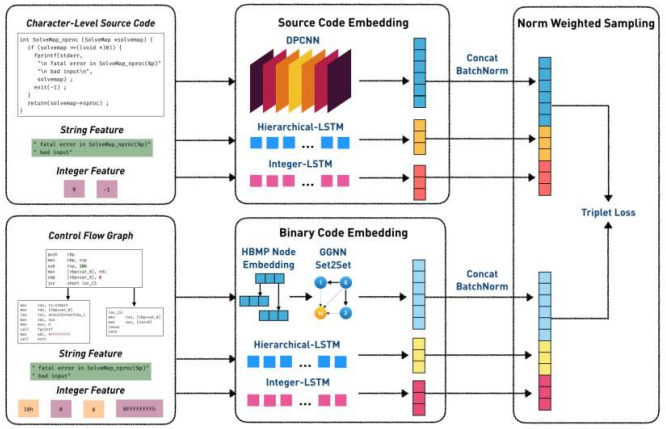
算法原理

- 源代码语义特征提取
  - DPCNN 深度金字塔卷积神经网络
  - 解决TextCNN 不能获得文本的长距离依赖关系问题



算法原理

- 字符串特征提取
  - 首先使用hierarchical-LSTM 得到每个字符串的向量
  - 然后使用sum pooling得到字符串集合的向量
- 立即数特征提取
  - Integer-LSTM
    - 输入：integer token；integer number
    - integer number作用在LSTM的输入门和输出门，从而控制信息流动



Binary:	fatal error in SolveMap_nproc(%p) bad input
Source:	fatal error in SolveMap_nproc(%p) bad input

$$f_k = \sigma(x_k W_f + h_{k-1} U_f + b_f)$$
$$i_k = \sigma(x_k W_i + h_{k-1} U_i + log(x_{kn}) \cdot N_i + b_i)$$
$$o_k = \sigma(x_k W_o + h_{k-1} U_o + log(x_{kn}) \cdot N_o + b_o)$$
$$c_k = f_k \odot c_{k-1} + i_k \odot \phi(x_k W_c + h_{k-1} U_c + b_c)$$
$$h_k = o_k \odot \phi(c_k)$$

实验结果

- 指标：recall@1/recall@10

Model	gcc-x64-O0		clang-arm-O3	
	Binary2Source	Source2Binary	Binary2Source	Source2Binary
BinPro	38.6 / 41.1	39.2 / 41.8	39.4 / 42.1	39.7 / 42.4
B2SFinder	34.1 / 39.6	34.4 / 39.8	33.5 / 39.2	34.2 / 39.5
TextCNN + HBMP	54.3 / 84.7	54.7 / 85.1	48.8 / 82.5	49.3 / 82.8
LSTM + HBMP	63.7 / 89.4	63.9 / 88.7	60.2 / 86.9	60.6 / 87.3
DPCNN + Word2vec	69.2 / 91.0	69.6 / 90.7	63.6 / 88.3	64.0 / 88.5
DPCNN + BERT	74.3 / 93.9	74.5 / 94.0	66.1 / 89.0	66.5 / 89.5
DPCNN + HBMP	80.8 / 96.4	81.2 / 96.6	72.9 / 91.2	73.2 / 92.1
Hungarian (Integer)	9.0 / 15.6	11.6 / 17.7	7.7 / 14.3	10.4 / 17.3
LSTM (Integer)	10.7 / 17.9	13.0 / 19.0	8.9 / 15.2	10.9 / 18.6
Integer-LSTM (Integer)	12.3 / 19.4	15.5 / 23.2	11.5 /17.1	12.2 / 20.7
Hungarian (String)	33.9 / 35.4	34.0 / 35.5	35.6 / 36.2	35.8 / 37.3
Hier-LSTM (String)	42.4 / 44.5	42.8 / 45.1	45.0 / 46.9	45.5 / 48.7
Random	81.9 / 97.3	82.3 / 98.0	74.2 / 92.0	74.8 / 92.6
Distance-Weight	86.2 / 97.4	86.5 / 97.8	77.4 / 94.3	78.2 / 94.7
Norm-Weight ($s = 0.5$)	85.3 / 97.2	85.4 / 97.5	76.9 / 93.4	77.5 / 93.6
Norm-Weight ($s = 2$)	89.0 / 97.9	89.1 / 98.2	81.2 / 95.1	82.5 / 95.4
Norm-Weight ($s = 5$)	**90.2 / 98.3**	**90.3 / 98.5**	**87.3 / 97.5**	**87.7 / 97.9**

- 源代码/二进制代码潜在语义特征重要
- 深度端到端模型比浅层预训练模型具有更优信息提取能力
- Integer-LSTM表现优于传统LSTM

总结

- 改进
  - 跨模态
    - 在获得最终向量前两个模态没有信息融合
      - 两个模态分别单独预训练
      - 跨语言模型的方法融合训练
  - 源代码特征提取
    - ASTNN
    - Tree-LSTM
  - 二进制代码特征提取
    - BERT
    - GCN

算法原理

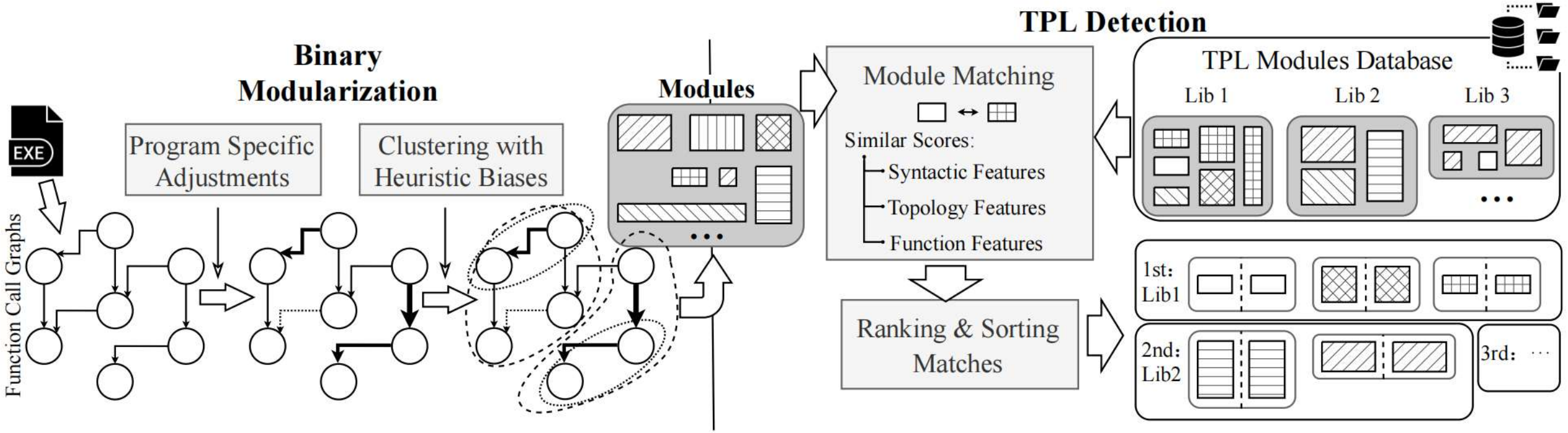
算法原理

T	二进制程序TPL检测
I	TPLs、待测二进制程序
P	1. 模块化技术 2. 模块相似性度量 3. TPL检测
O	匹配结果

P	将TPL作为整体建模，难以应用在程序只导入部分库的场景
C	不同版本TPL功能未发生显著变化
D	程序/TPL模块化技术
L	2022 ISCE（软件工程顶会）

算法原理

- 算法原理图


Binary
Modularization
Program Specific
Adjustments
Clustering with
Heuristic Biases
EXE
Function Call Graphs
Modules
TPL Detection
Module Matching
Similar Scores:
Syntactic Features
Topology Features
Function Features
Ranking & Sorting
Matches
TPL Modules Database
Lib 1
Lib 2
Lib 3
1st:
Lib1
2nd:
Lib2
3rd:

算法原理

- 模块化技术
  - 问题1：如何对程序/库进行模块化？
    - 模块化算法
  - 问题2：如何衡量模块化的质量？
    - 模块化质量评估

- 动机：具有相似功能的函数很可能彼此靠近，从而在函数调用图中形成“社区”

- 模块化质量评估
  - 基线+微调
    - 基线 基于社区检测算法指标
    - 调整 根据程序特殊性对指标调整

算法原理

- 模块化质量评估
  - 模块总体质量分数Q（Girvan Newman 算法）
    - 物理意义：
      - Q越大，模块内函数一致性越高
      - 如果函数i和j属于同一模块，且相互连接，Q将会增加
      - 如果函数i和j属于同一模块，但没有连接，Q将会降低
    - 参数含义
      - i, j：函数
      - m: 图中的边数
      - A_{ij}：函数是否相互连接
      - k_{ij}： 节点的入度和出度
      - $\delta(Ci, Cj)$： 函数是否属于同一模块

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(C_i, C_j)$$

算法原理

- 模块化质量评估
  - 根据函数体积对指标进行调整
    - 动机：对于具有层次结构的程序，顶层的函数倾向于控制，通过调用来实现的底层函数的行为
    - 目的：根据函数体积和连通性为每个函数分配不同的权重；顶层 > 底层

$$FV'(u) = FV(u) + c \sum_{v \in E(u)} \frac{FV(v)}{C_v}$$

  - 最终指标

$$Q = \frac{1}{2W} \sum_{i,j} [w_{ij} - \frac{k_i^{out} k_j^{in}}{2W}] \delta(C_i, C_j)$$

算法原理

- 模块化算法
  - 基于社区检测算法和2个偏差项
    - Fast Unfolding Louvain Algorithm
      - 首先将调用图中每个函数视为单个模块
      - 将模块和邻居模块尝试融合，使Q增加

$$\Delta Q_{r,s} = e_{r,s}^{in} + e_{r,s}^{out} + e_{s,r}^{in} + e_{s,r}^{out} - 2 * (a_r^{in} * a_s^{in} + a_r^{out} * a_s^{out})$$

$$e_{r,s}^{in} = \sum_{i \in r} \sum_{j \in s} \frac{k_i^{in} k_j^{out}}{2W}; \quad e_{r,s}^{out} = \sum_{i \in r} \sum_{j \in s} \frac{k_i^{out} k_j^{in}}{2W}$$

$$a_r^{in} = \sum_s e_{s,r} \delta(r,s); \quad a_r^{out} = \sum_s e_{r,s} \delta(r,s)$$

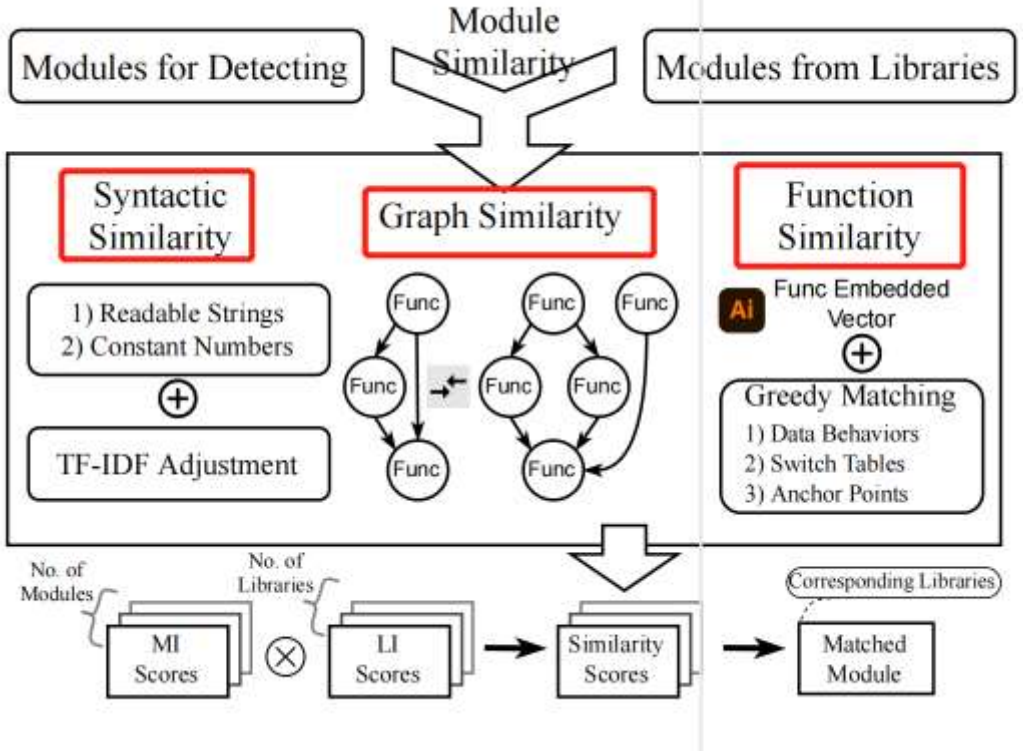
算法原理

- 模块化算法
  - 局部偏差
    - 目标：功能相近的函数分组至一个模块
    - 为函数在二进制文件中的位置分配索引号，根据模块的函数索引平均值，让算法首先考虑彼此索引接近的函数
  - 模块合并限制偏差
    - 目标：每个模块封装后功能单一

$$\Delta Q = \Delta Q' \times B_l \times B_e$$

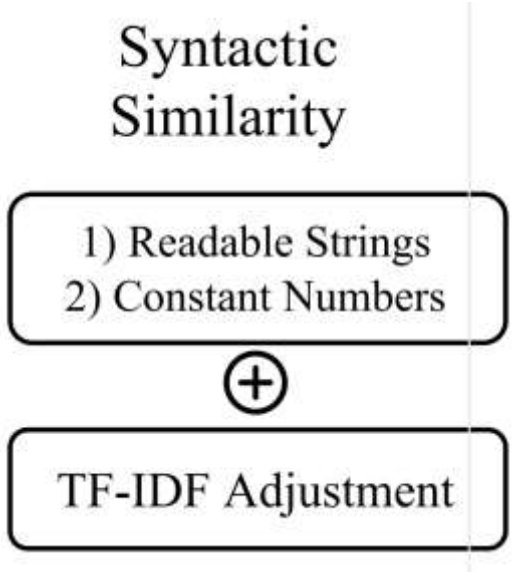
算法原理

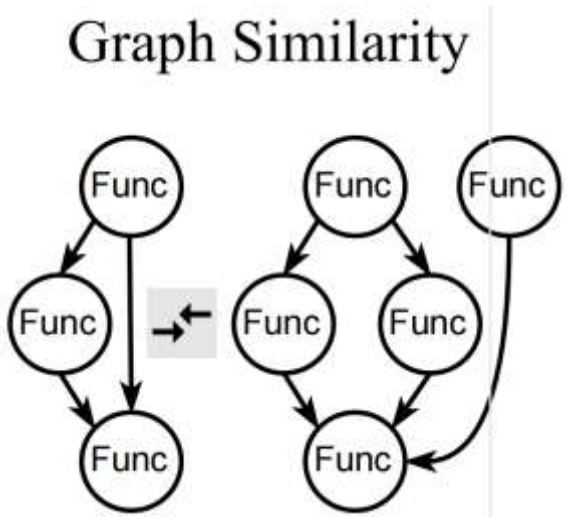
- 模块相似性度量
  - 语法相似特征（Syntactic Features）
  - 图相似性特征（Graph Similarity Features）
  - 函数相似特征（Function Similarity Features）



算法原理

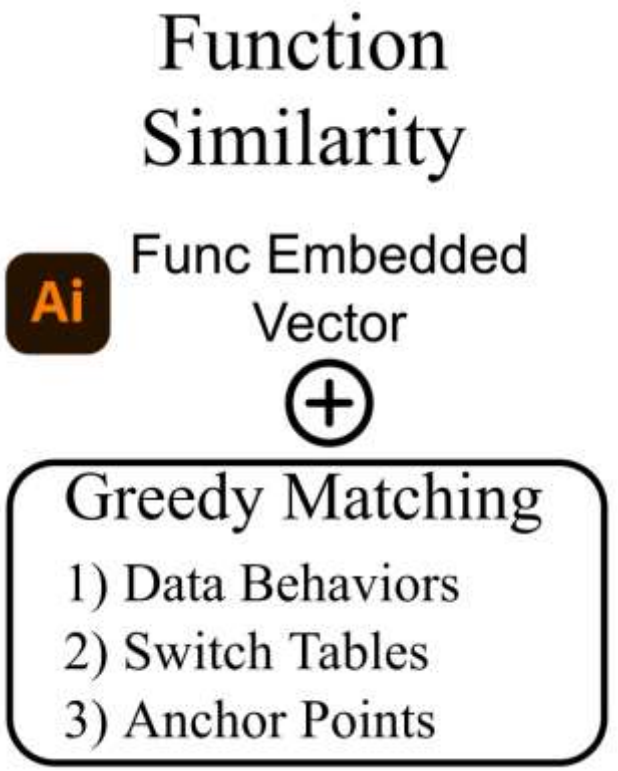
- 模块相似性度量
  - 语法相似特征（Syntactic Features）
    - 字符串
    - 数字：使用Tf-Idf为稀有常数分配更多权重
  - 调用图相似特征（Graph Similarity Features）
    - propagation graph kernel algorithm：度量图和子图相似度





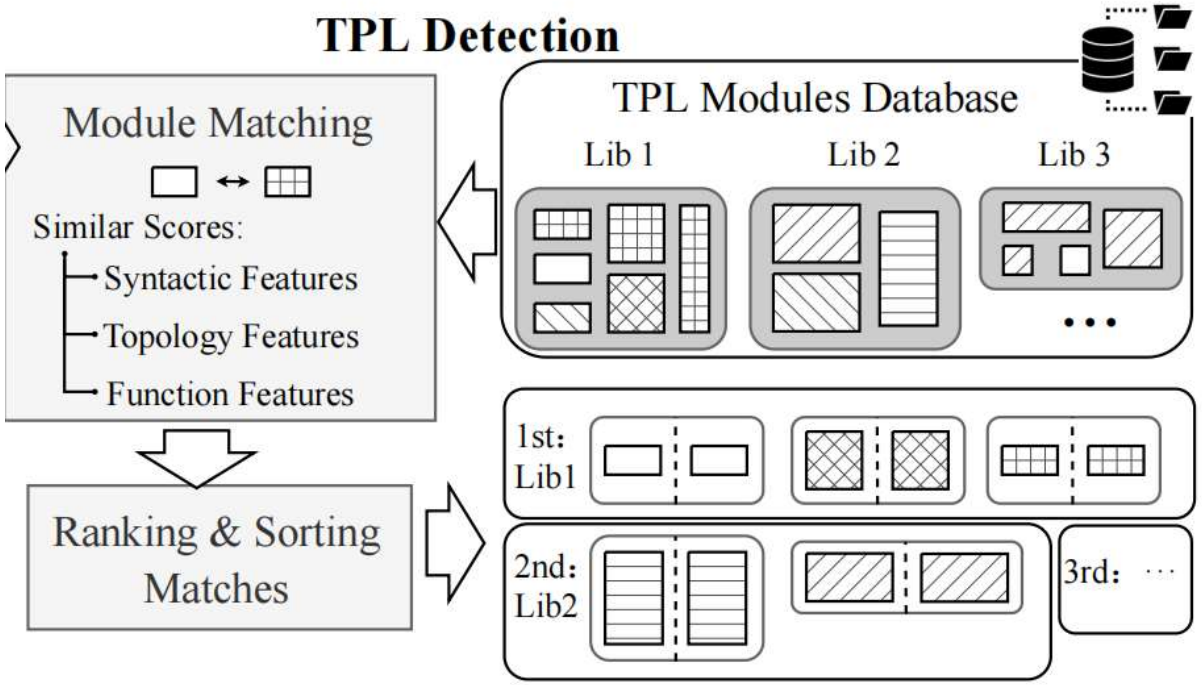
算法原理

- 模块相似性度量
  - 函数相似特征（Function Similarity Features）
    - 问题1：如何度量函数相似性?
      - 二进制程序函数相似性度量（Gemini）
    - 问题2：如何选择计算相似度的两个函数?
      - 在其他模块中检测类似锚点并选择函数
      - 锚点
        - » 在同一内存空间中访问数据的函数
        - » 访问函数调度表：指针或内存地址表



算法原理

- TPL检测
  - 整体步骤：
    - 匹配目标程序和TPL中的模块，根据相似度得分对候选库进行排序
  - 假阳性原因：
    - 一些库可能包含相似功能模块
    - TPL的模块数不同
      - Libbz2 library
        - » 5个模块
        - » 81个函数
      - Libcrypto library
        - » 186个模块
        - » 6559个函数



算法原理

- 匹配置信度
  - 模块重要性分数 (MI)
    - 模块规模越大，模块越重要
  - 库重要性分数 (LI)
    - 与参考频率呈正相关
    - 与包含的模块的数量呈负相关

$$MI_k = \frac{|m_k|}{\sum_i^n |m_i|/n}$$

$$LI_h = \frac{\log(v(l_h)+1)}{|l_h|}$$

$$MC_k = MI_k \times LI_h$$

实验结果

- Real-world Programs

	MODX	OssPolice (1)	OssPolice (2)	BAT(1)	BAT(1)
Package Detection					
Precision(%)	83.0	83.8	82	66.1	75
Recall(%)	73.8	70.0	87	65.7	61
Library Detection					
Precision(%)	85.6	77.8	/	41.4	/
Recall(%)	49.6	40.2	/	38.7	/

- 准确率高，模块由执行类似功能的函数组成
- 语义信息增加模块之间的差异

实验结果

- Partial Library Detection

Binary	Libs Linked	ModX			OssPolice			BAT		
		TP	FP	FN	TP	FP	FN	TP	FP	FN
ssldump	2	2	0	0	2	0	0	2	2	0
vim	4	2	0	2	1	0	3	1	3	3
busybox	3	1	1	2	1	0	2	1	4	2
tcpdump	3	3	0	0	3	0	0	2	1	1
openvpn	5	4	0	1	3	2	2	3	1	2
sqlite3	4	3	1	1	2	2	2	2	2	2
openssl	5	2	1	3	3	2	2	3	1	2
Total	26	17	3	9	15	6	11	14	14	12

Performance Summary

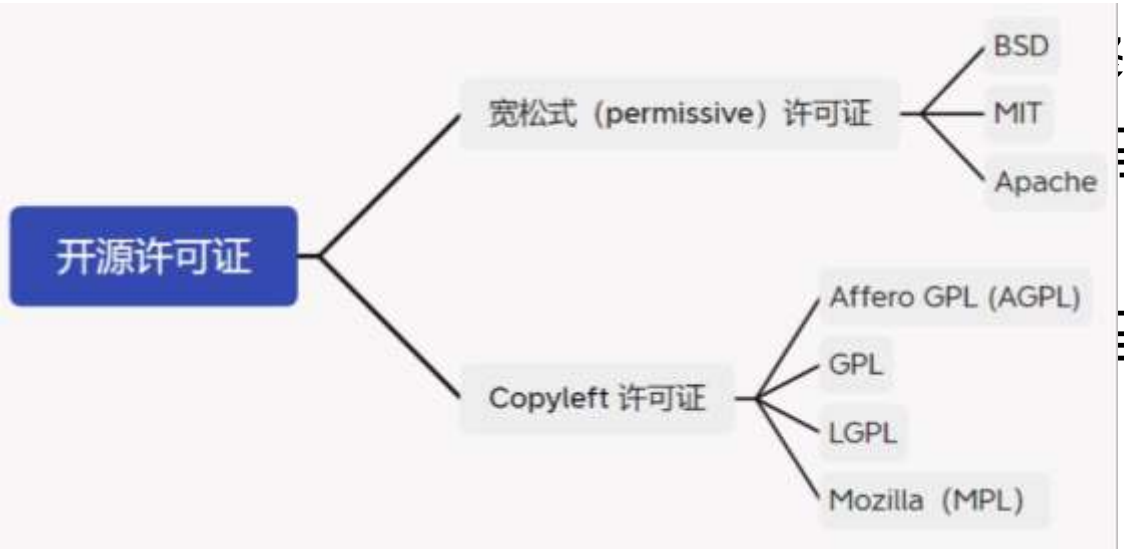
ModX		OssPolice		BAT	
Precision	Recall	Precision	Recall	Precision	Recall
85.0%	65.4%	71.4%	57.7%	50%	53.8%

- 导入部分库时，模块特性保持而稳定不会被破坏
- 难以从规模小的库中提取可区分特性

应用总结

应用总结

- 代码抄袭检测
- 漏洞同源性判别
- 违反开源许可证分析
  - 2021年12月，抖音下架TikTok Live Studio，该APP违反GPL许可证
- 开源许可证
  - 开源许可证是一种针对开源软件使用者的约束，目的在于规范受著作权保护的软件的使用或者分发行为。
  - GPL许可证——
  - MPL许可证——



参考文献

[1] Yang C, Xu Z, Chen H, et al. ModX: Binary Level Partially Imported Third-Party Library Detection via Program Modularization and Semantic Matching[C]//2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE). IEEE, 2022: 1393-1405.

[2] Yu Z, Zheng W, Wang J, et al. Codecmr: Cross-modal retrieval for function-level binary source code matching[J]. Advances in Neural Information Processing Systems, 2020, 33: 3872-3883.

谢谢！

大成若缺，其用不弊。大盈若冲，其用不穷。大直若屈。大巧若拙。大辩若讷。静胜躁，寒胜热。清静为天下正。