Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

基于深度学习的二进制软件漏洞挖掘

硕士研究生　闫晗

2019年12月15日

内容提要

- 背景简介
- 基本概念
- 算法原理
- 优劣分析
- 应用总结
- 参考文献

背景简介

- 预期收获
  - 1. 了解二进制漏洞挖掘任务的基本概念和TIPO
  - 2. 了解二进制缺陷检测任务的基本概念和TIPO
  - 3. 了解深度学习与二进制缺陷检测任务的常见结合方式
  - 4. 理解一种基于深度学习的二进制缺陷检测方法的原理

基本概念

基本概念

- 漏洞
  - 软件漏洞 (Vulnerablity) 是由软件在设计、开发或配置过程中存在的错误 (Mistake) 而导致的缺陷 (Flaw) 的实例，利用漏洞可以违反某些显式或隐式的安全策略。


```
//STACK_OVER_FLOW
void stack_over_flow(unsigned char* data)
{
    //缓冲区
    unsigned char buffer[BUF_LEN];

    //拷贝数据至缓冲区
    strcpy((char*)buffer, (char*)data);
}
```

错误：拷贝动作没有约束边界

缺陷：可能导致栈缓冲区溢出

漏洞：可能导致恶意代码执行

基本概念

- 漏洞挖掘

T	发现目标软件中存在的漏洞
I	目标软件（源程序或二进制程序）
P	1. 软件缺陷检测（缺陷位置）； 2. 可利用性分析（利用方法）
O	目标软件的漏洞信息（缺陷位置、利用方法等）

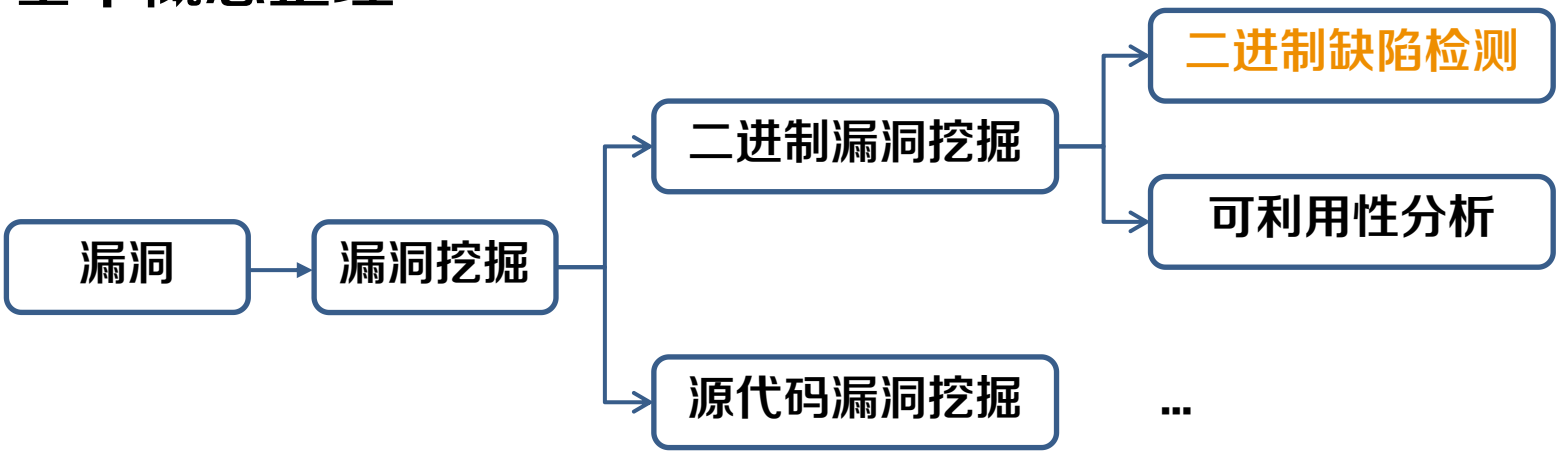
- 二进制漏洞挖掘
  - 目标软件是二进制文件的漏洞挖掘
  - 二进制文件包括可执行文件、链接库文件等

基本概念

- 二进制缺陷检测

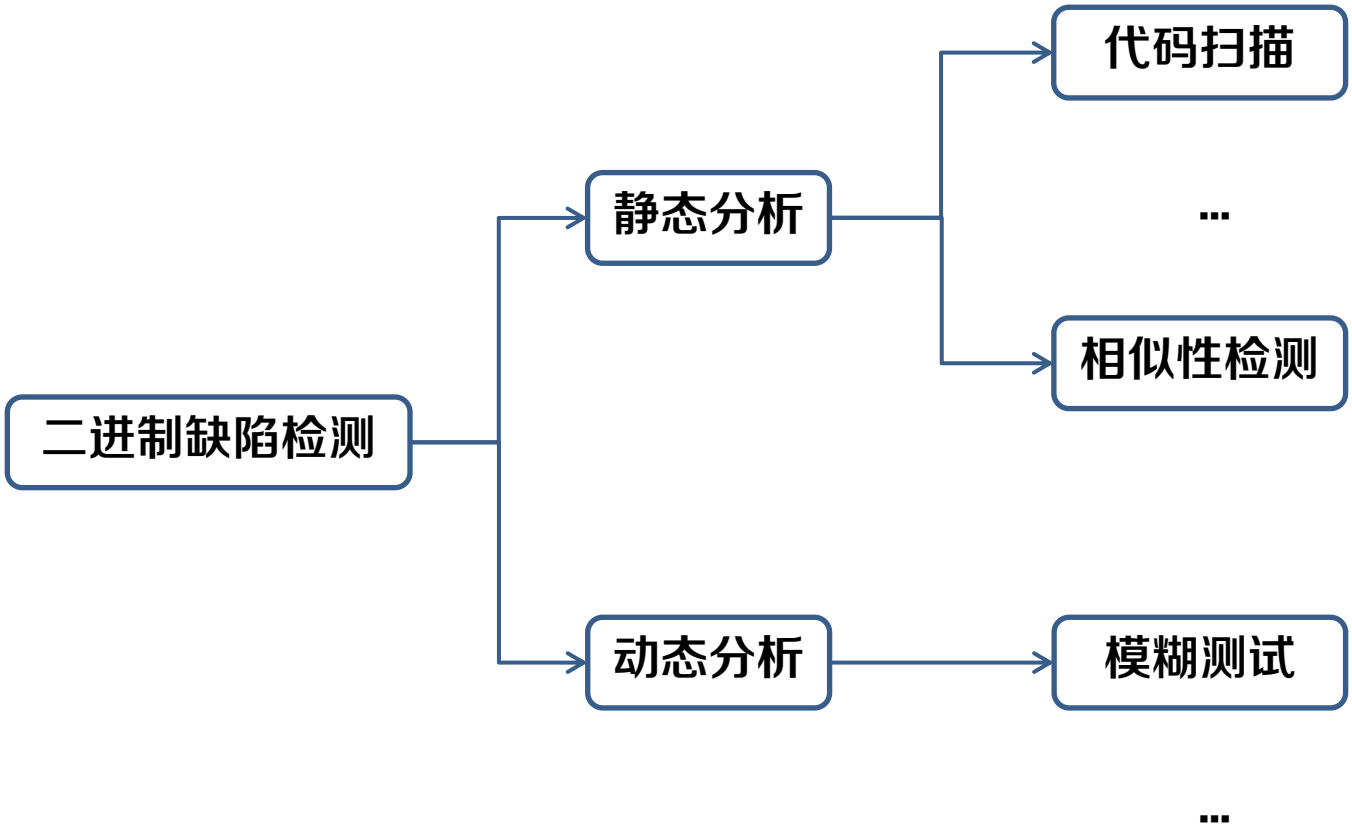
T	发现并定位二进制程序中存在的缺陷代码
I	目标二进制程序
P	静态分析/动态分析
O	目标二进制程序的缺陷代码位置

- 基本概念整理



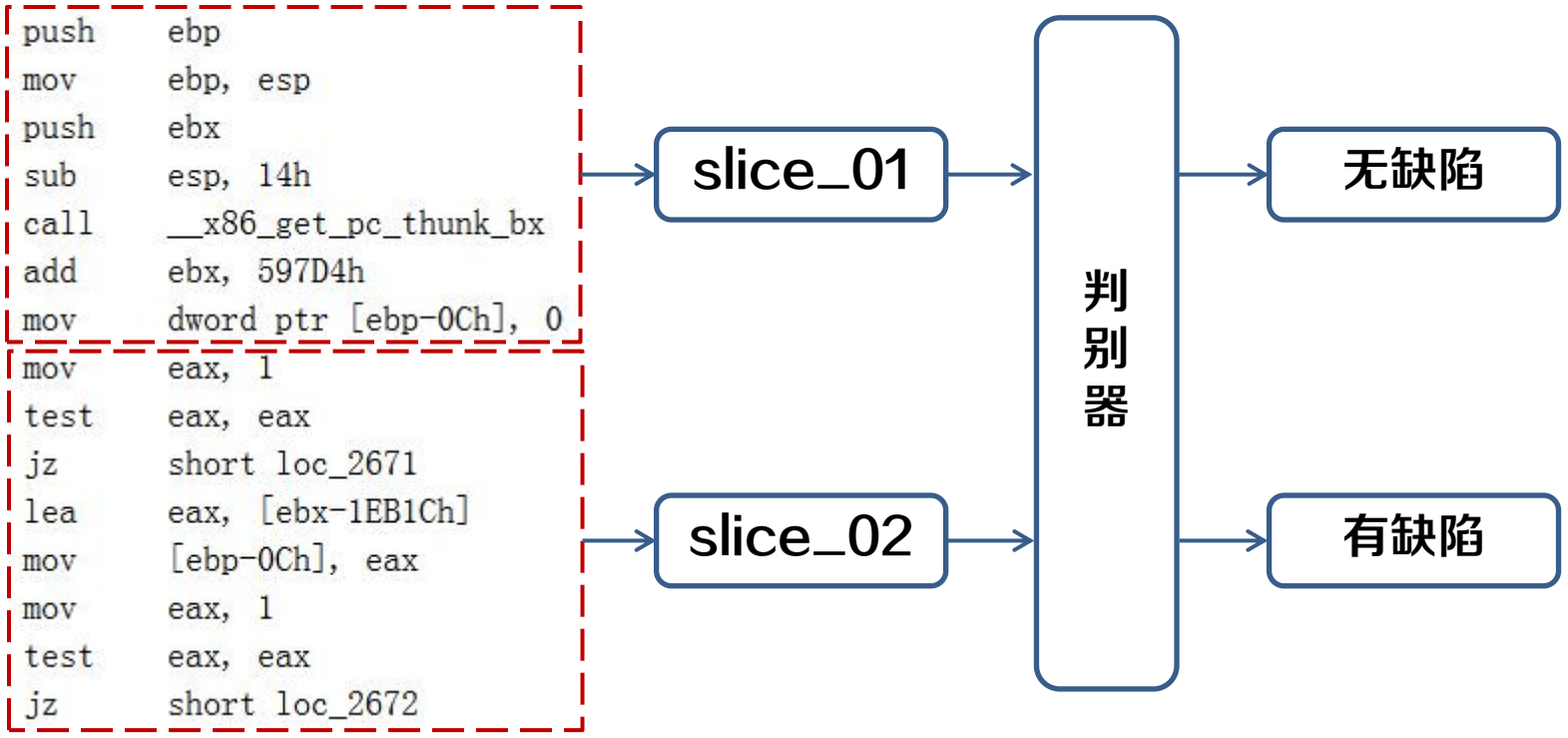
基本概念

- 二进制缺陷检测
  - 主要技术和方法



基本概念

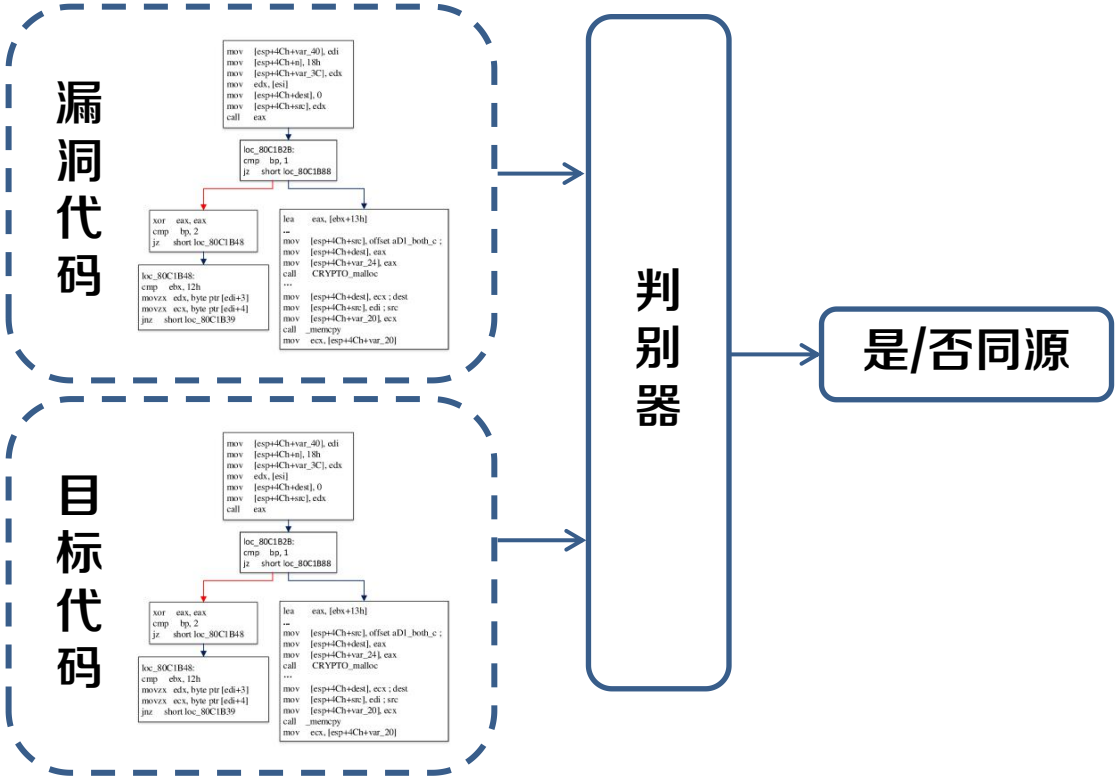
- 二进制缺陷检测（代码扫描）
  - 基本概念：对汇编代码进行遍历切片并判断切片是否有缺陷



  - 经典方法：基于规则匹配的代码扫描

基本概念

- 二进制缺陷检测（相似性检测）
  - 基本概念：匹配与已知漏洞代码同源的代码



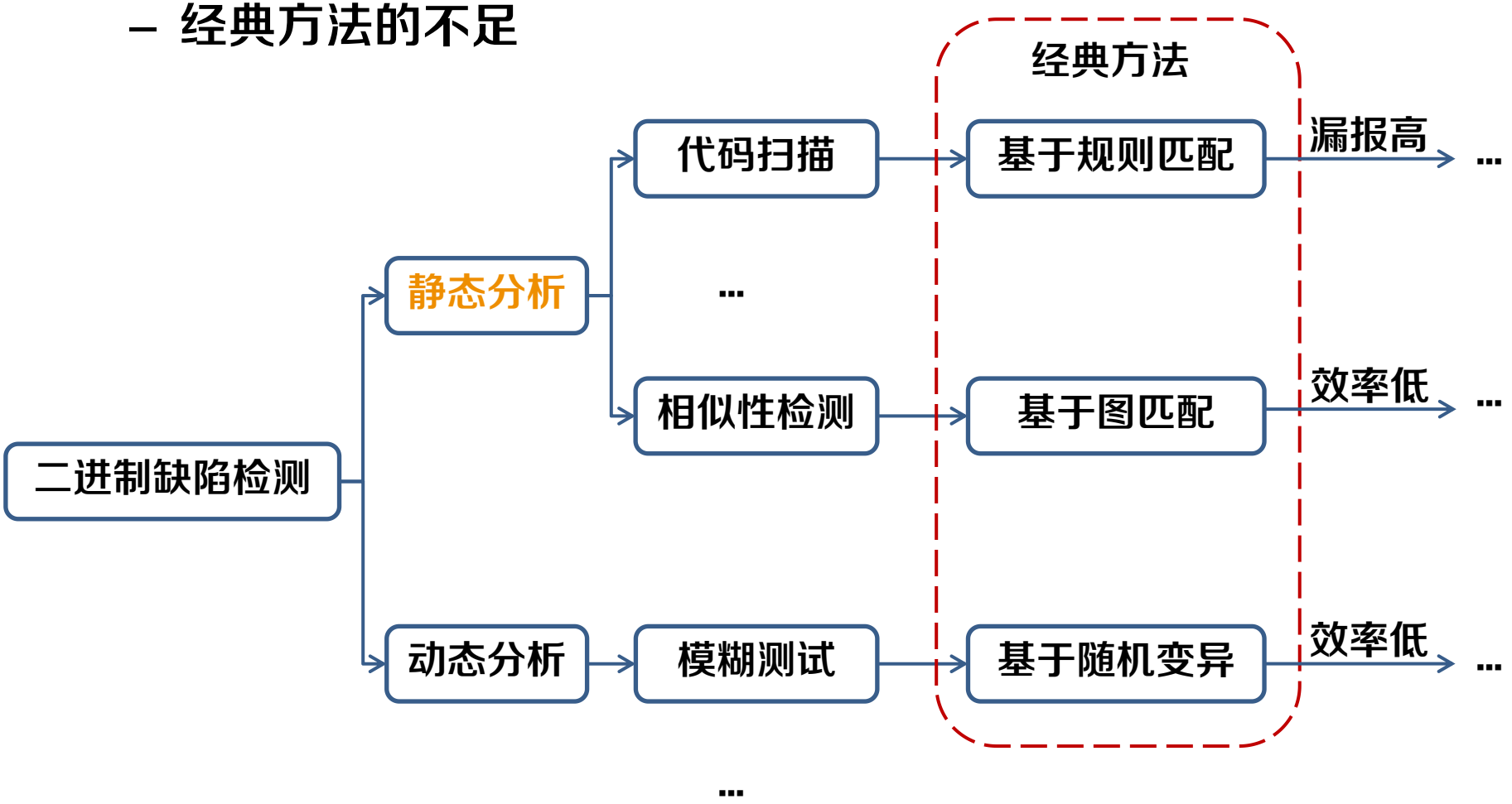
  - 经典方法：基于图匹配(Graph Matching)的相似性检测

基本概念

- 二进制缺陷检测（模糊测试）
  - 基本概念：通过构造随机的、非预期的畸形数据作为程序的输入，并监控程序执行过程中可能产生的异常，之后将这些异常作为分析的起点，确定其可利用性。
  - 经典方法：基于随机变异的模糊测试

基本概念

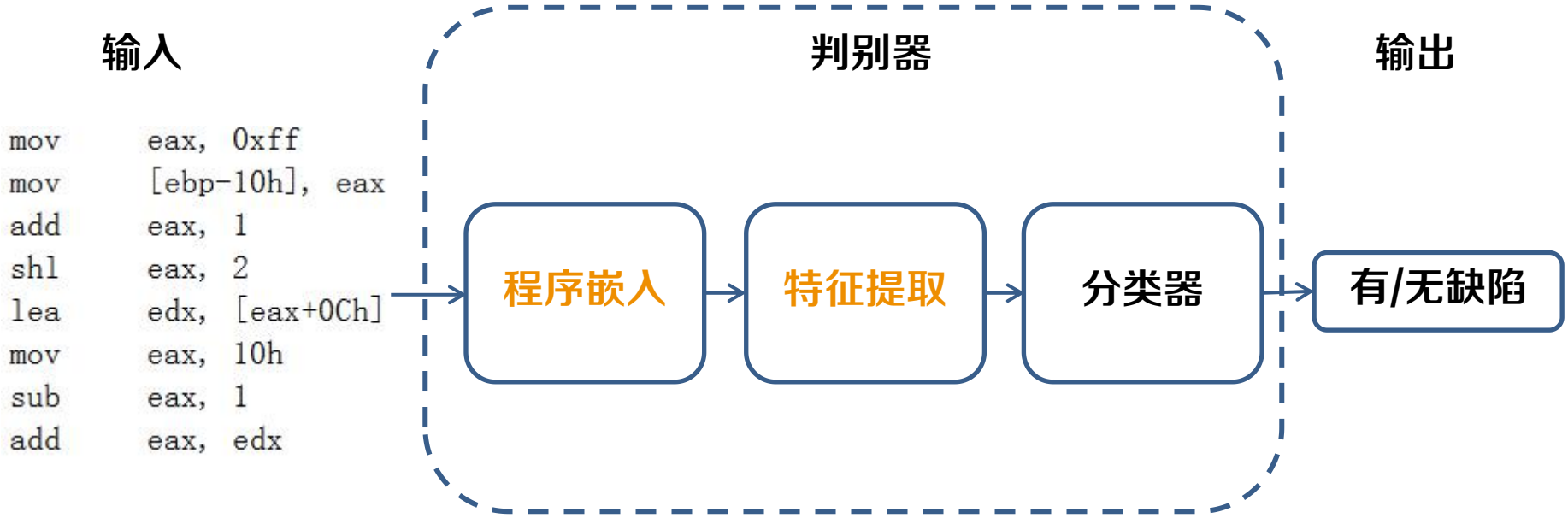
- 二进制缺陷检测
  - 经典方法的不足



算法原理

算法原理

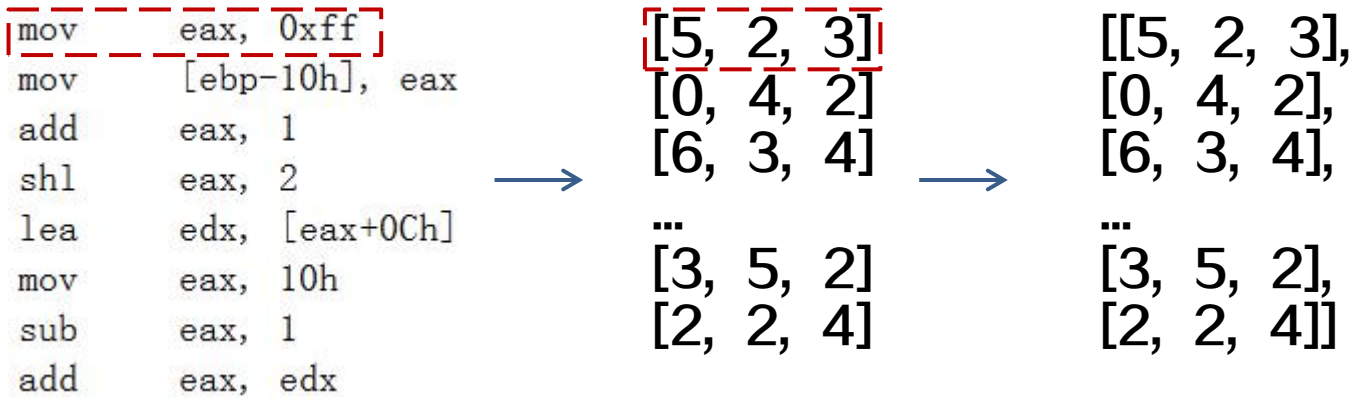
- 基于深度学习的代码扫描
  - 总体框架



slice

算法原理

- 基于深度学习的代码扫描
  - 程序嵌入（Program Embedding）模块
    - 目标：将程序（在静态分析中通常以汇编代码表示）嵌入到向量空间中
    - 方法：目前比较常见的做法是，首先将每个指令单独嵌入到向量空间中，然后对指令的向量表示进行组合等处理，从而得到程序的向量表示



算法原理

- 基于深度学习的代码扫描
  - 程序嵌入（Program Embedding）模块
    - 研究问题：如何将指令嵌入到向量空间
    - 方法1：将每一条汇编指令视为一个词（word），利用word2vec算法将指令嵌入到向量空间中
    - 缺点1：指令空间是无限的，导致word2vec的词表过大
    - 缺点2：指令被视为独立的整体，具有相似结构的指令可能不具有相近的向量表示

```
mov     eax, 0xff
mov     [ebp-10h], eax
add     eax, 1
shl     eax, 2
lea     edx, [eax+0Ch]
mov     eax, 10h
sub     eax, 1
add     eax, edx
```

word_1

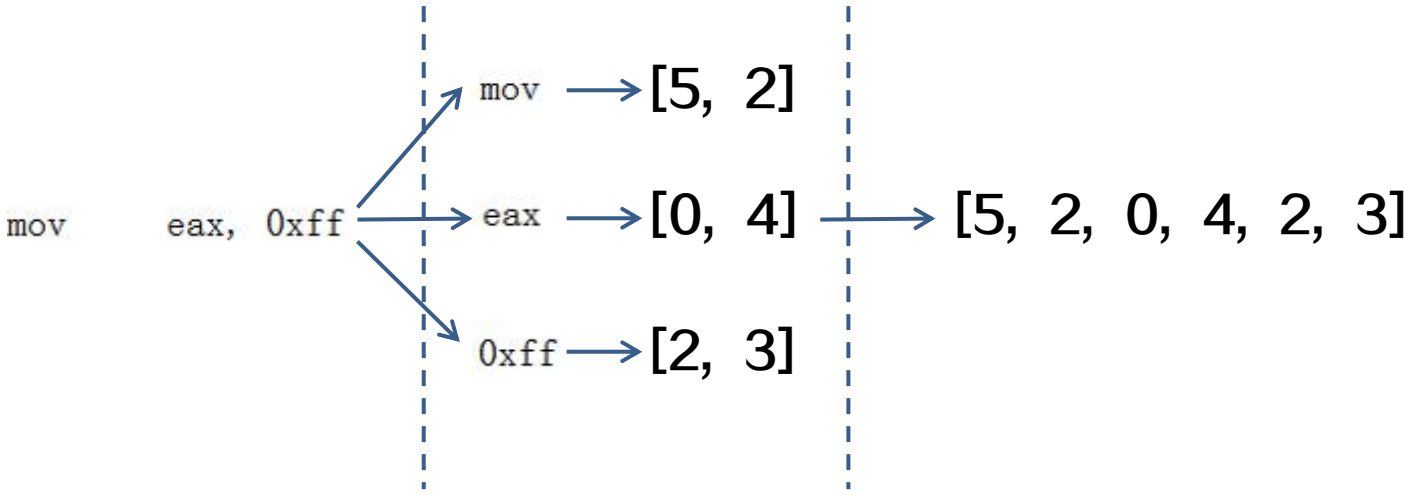
word_2

```
mov     ebx, 0xff
```

如何表示？

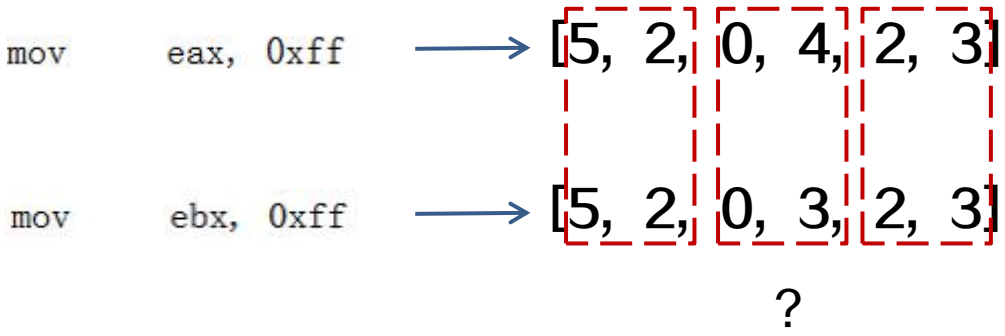
算法原理

- 基于深度学习的代码扫描
  - 程序嵌入（Program Embedding）模块
    - 研究问题：如何将指令嵌入到向量空间
    - 方法2：将每一条汇编指令视为一个句子（sentence），而将汇编指令的元素（操作码等）视为词（word），首先嵌入词向量，其后将词向量按序拼接形成句子向量，也就是指令的向量表示



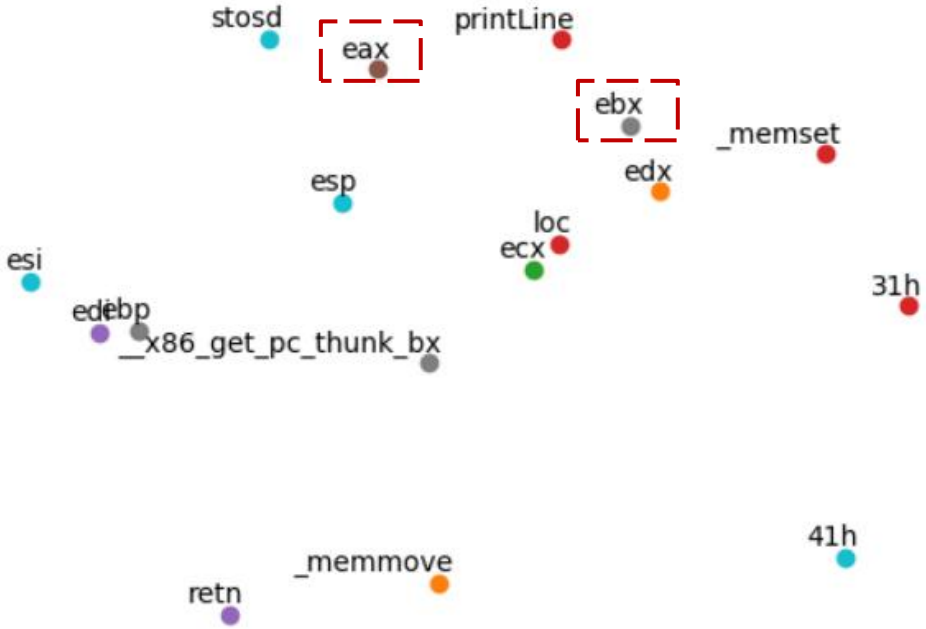
算法原理

- 基于深度学习的代码扫描
  - 程序嵌入（Embedding）模块
    - 研究问题：如何将指令嵌入到向量空间
    - 方法2
    - 优势1：对于汇编指令而言，元素的种类很少，因此使用有限的词表来表示无限的指令空间
    - 优势2：相似的指令具有相近的向量表示



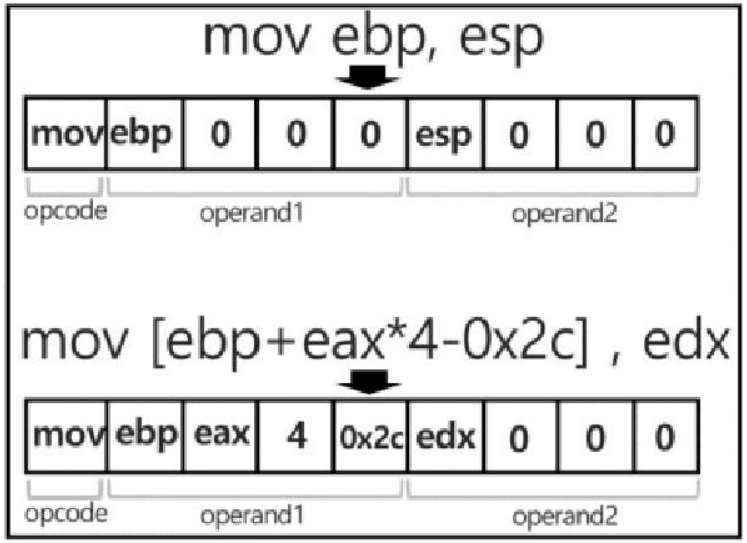
算法原理

- 基于深度学习的代码扫描
  - 程序嵌入（Embedding）模块
    - 研究问题：如何将指令嵌入到向量空间
    - 方法2
    - 优势2：相似的指令具有相近的向量表示



算法原理

- 基于深度学习的代码扫描
  - 程序嵌入（Embedding）模块
    - 研究问题：如何将指令嵌入到向量空间
    - 方法2
    - 元素划分：instruction = 1 + 4 + 4



操作数：基址 + 偏移地址

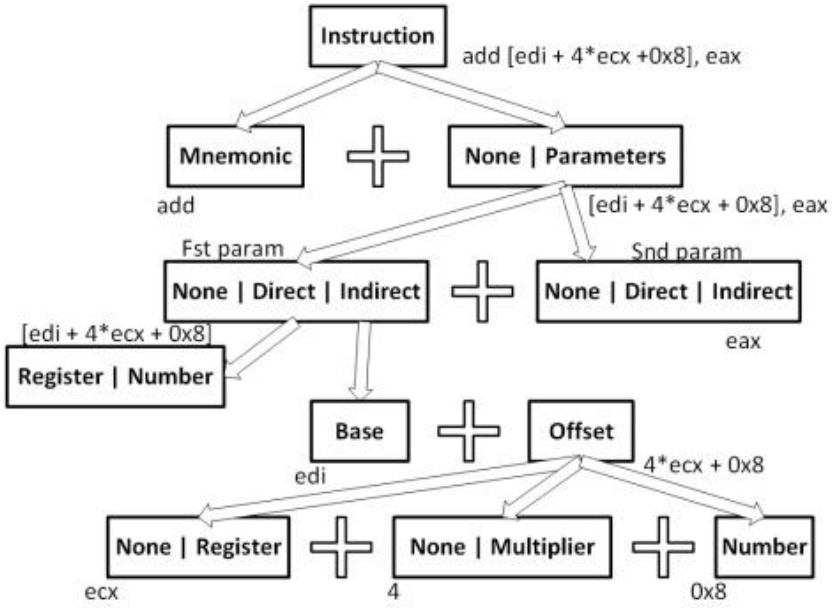
SPR：ESP EBP ESI EDI

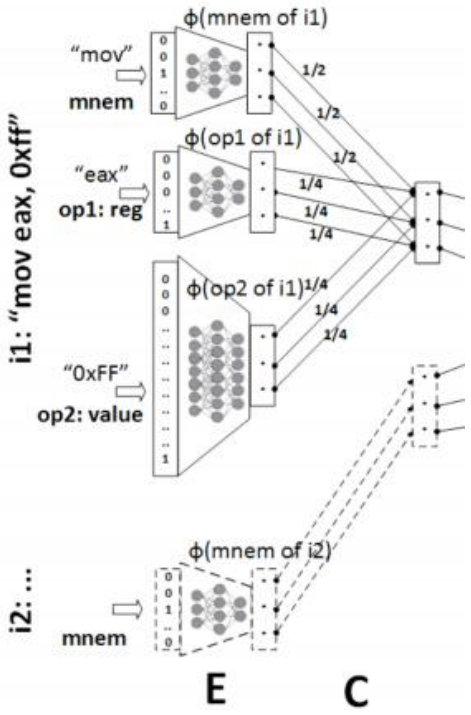
GPR：EAX EBX ECX EDX

VALUE：HEX DEC

算法原理

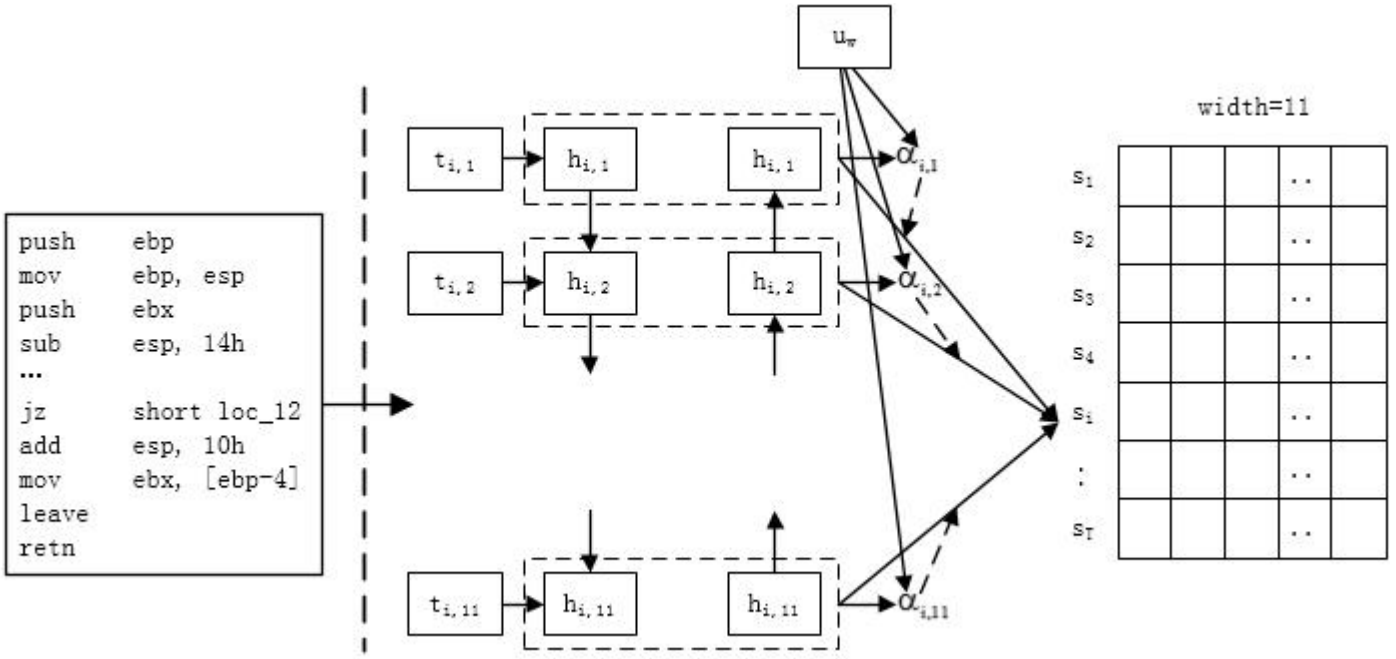
- 基于深度学习的代码扫描
  - 程序嵌入（Embedding）模块
    - 研究问题：如何将指令嵌入到向量空间
    - 方法3：将汇编指令视为句子，通过词向量的加权和构成句子向量，其中，权值由元素的层级决定





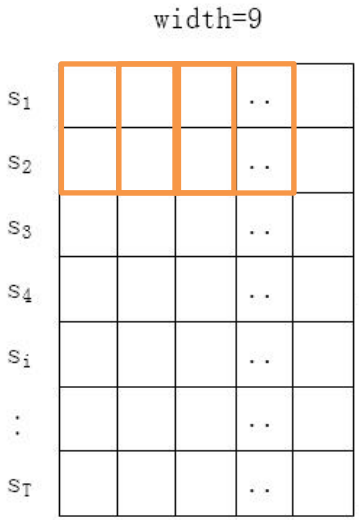
算法原理

- 基于深度学习的代码扫描
  - 程序嵌入（Embedding）模块
    - 研究问题：如何将指令嵌入到向量空间
    - 方法4：将汇编指令视为句子，通过神经网络学习句子向量
    - 该方法仅供参考



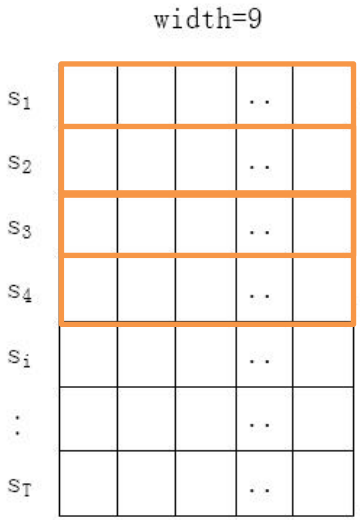
算法原理

- 基于深度学习的代码扫描
  - 特征提取模块
    - 研究问题：如何选择合适的特征提取模型
    - 物理背景：缺陷代码一般具有局部性特征
    - 模型1：CNN
    - 缺点：2-D卷积对学习文本数据无意义，导致低准确率



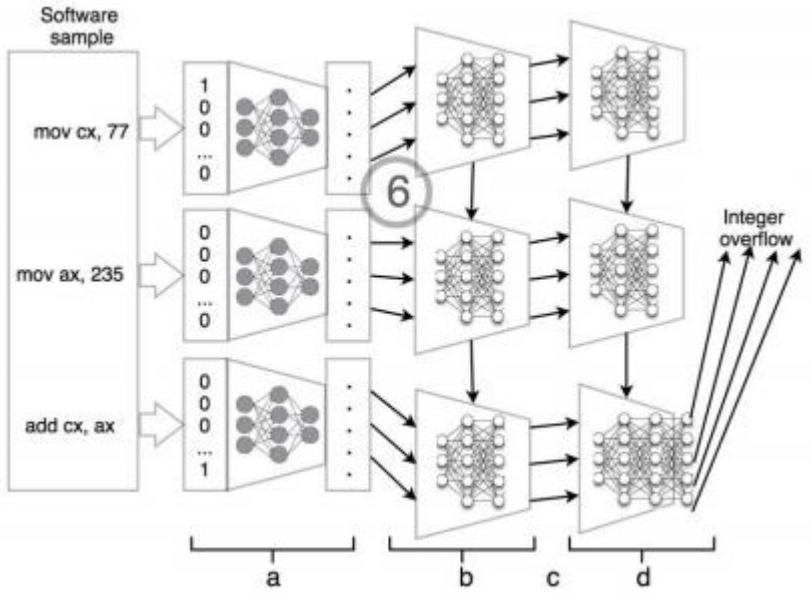
算法原理

- 基于深度学习的代码扫描
  - 特征提取模块
    - 研究问题：如何选择合适的特征提取模型
    - 物理背景：缺陷代码一般具有局部性特征
    - 模型2：Text-CNN
    - 优点：1-D卷积适合捕捉文本的局部特征



算法原理

- 基于深度学习的代码扫描
  - 特征提取模块
    - 研究问题：如何选择合适的特征提取模型
    - 物理背景：缺陷代码一般具有局部性特征
    - 模型3：Bi-LSTM



算法原理

- 基于深度学习的代码扫描
  - 实验结果
    - 方法：词向量拼接 + TextCNN
    - 数据集1：NIST Juliet TestSuite for C/C++

Description	Accuracy	Recall	Precision
Instruction2vec	96.81%	97.07%	96.65%
Binary2img	97.53%	97.05%	97.91%
Word2vec	96.01%	96.07%	95.92%

    - 数据集2：IARPA STONESOUP

Description	Accuracy	Loss	Recall	Precision
Instruction2vec	91.11%	0.3058	93.33%	70.00%
Binary2img	53.33%	0.6894	46.39%	49.67%
Word2vec	18.98%	4.6809	100%	18.98%

算法原理

- 基于深度学习的代码扫描
  - 实验复现
    - 数据准备
      - Juliet TestSuite for C/C++ v1.3 - CWE121
      - 使用测试套件中提供的MakeFile编译可执行文件
      - 使用IDA Pro反汇编得到并导出可执行文件
      - 提取有/无缺陷的函数（或切片）
    - 预处理
      - 编写分词脚本，对汇编指令进行分词
      - 将所有分词得到的元素统一写入csv，做word2vec
    - 分类
      - 加载数据集，并根据word2vec字典将元素数值化
      - 训练、测试（多数经典模型的代码开源）

算法原理

- 基于深度学习的代码扫描
  - 实验复现
    - 数据加载

```
push    ebp
mov     ebp, esp
push    ebx
sub     esp, 14h
call    __x86_get_pc_thunk_bx
add     ebx, 597D4h
mov     dword ptr [ebp-0Ch], 0
mov     eax, 1
test    eax, eax
jz      short loc
lea     eax, [ebx-1EB1Ch]
mov     [ebp-0Ch], eax
```

```
['push', 'REG', 'NULL', 'ebp', 'PH', 'PH', 'PH', 'PH', 'PH', 'PH', 'PH']
['mov', 'REG', 'REG', 'ebp', 'PH', 'PH', 'PH', 'esp', 'PH', 'PH', 'PH']
['push', 'REG', 'NULL', 'PH', 'ebx', 'PH', 'PH', 'PH', 'PH', 'PH', 'PH']
['sub', 'REG', 'IMM', 'esp', 'PH', 'PH', 'PH', 'PH', 'PH', 'PH', '14h']
['call', 'MARK', 'NULL', 'PH', 'PH', 'PH', '__x86_get_pc_thunk_bx', 'PH', 'PH', 'PH', 'PH']
['add', 'REG', 'IMM', 'PH', 'ebx', 'PH', 'PH', 'PH', 'PH', 'PH', '597D4h']
['mov_dword_ptr', 'MEM', 'IMM', 'ebp', 'PH', 'PH', '0Ch', 'PH', 'PH', 'PH', '0']
['mov', 'REG', 'IMM', 'PH', 'eax', 'PH', 'PH', 'PH', 'PH', 'PH', '1']
['test', 'REG', 'REG', 'PH', 'eax', 'PH', 'PH', 'PH', 'eax', 'PH', 'PH']
['jz_short', 'MARK', 'NULL', 'PH', 'PH', 'PH', 'loc', 'PH', 'PH', 'PH', 'PH']
['lea', 'REG', 'MEM', 'PH', 'eax', 'PH', 'PH', 'PH', 'ebx', 'PH', '1EB1Ch']
['mov', 'MEM', 'REG', 'ebp', 'PH', 'PH', '0Ch', 'PH', 'eax', 'PH', 'PH']
```

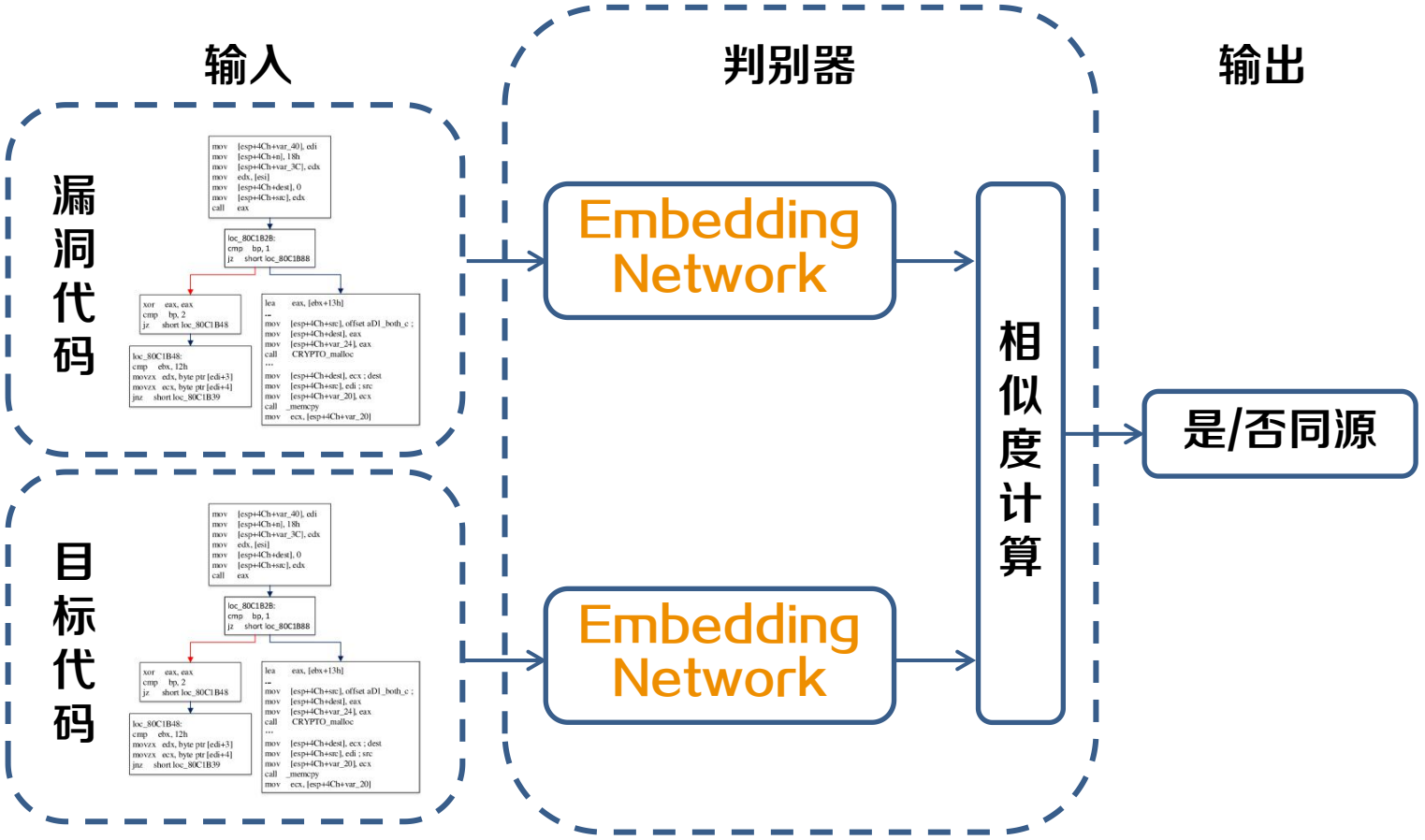
算法原理

- 基于深度学习的代码扫描
  - 实验复现
    - 数据加载

```
['push', 'REG', 'NULL', 'ebp', 'PH', 'PH', 'PH', 'PH', 'PH', 'PH', 'PH']                    [12, 2, 3, 7, 1, 1, 1, 1, 1, 1, 1]
['mov', 'REG', 'REG', 'ebp', 'PH', 'PH', 'PH', 'esp', 'PH', 'PH', 'PH']                     [8, 2, 2, 7, 1, 1, 1, 9, 1, 1, 1]
['push', 'REG', 'NULL', 'PH', 'ebx', 'PH', 'PH', 'PH', 'PH', 'PH', 'PH']                    [12, 2, 3, 1, 17, 1, 1, 1, 1, 1, 1]
['sub', 'REG', 'IMM', 'esp', 'PH', 'PH', 'PH', 'PH', 'PH', 'PH', '14h']                     [14, 2, 4, 9, 1, 1, 1, 1, 1, 1, 35]
['call', 'MARK', 'NULL', 'PH', 'PH', 'PH', '__x86_get_pc_thunk_bx', 'PH', 'PH', 'PH', 'PH'] [16, 11, 3, 1, 1, 1, 37, 1, 1, 1, 1]
['add', 'REG', 'IMM', 'PH', 'ebx', 'PH', 'PH', 'PH', 'PH', 'PH', '597D4h']                  [13, 2, 4, 1, 17, 1, 1, 1, 1, 1, 4021]
['mov_dword_ptr', 'MEM', 'IMM', 'ebp', 'PH', 'PH', '0Ch', 'PH', 'PH', 'PH', '0']            [27, 6, 4, 7, 1, 1, 15, 1, 1, 1, 20]
['mov', 'REG', 'IMM', 'PH', 'eax', 'PH', 'PH', 'PH', 'PH', 'PH', '1']                       [8, 2, 4, 1, 5, 1, 1, 1, 1, 1, 28]
['test', 'REG', 'REG', 'PH', 'eax', 'PH', 'PH', 'PH', 'eax', 'PH', 'PH']                    [50, 2, 2, 1, 5, 1, 1, 1, 5, 1, 1]
['jz_short', 'MARK', 'NULL', 'PH', 'PH', 'PH', 'loc', 'PH', 'PH', 'PH', 'PH']               [45, 11, 3, 1, 1, 1, 23, 1, 1, 1, 1]
['lea', 'REG', 'MEM', 'PH', 'eax', 'PH', 'PH', 'PH', 'ebx', 'PH', '1EB1Ch']                 [18, 2, 6, 1, 5, 1, 1, 1, 17, 1, 2781]
['mov', 'MEM', 'REG', 'ebp', 'PH', 'PH', '0Ch', 'PH', 'eax', 'PH', 'PH']                    [8, 6, 2, 7, 1, 1, 15, 1, 5, 1, 1]
```

算法原理

- 基于深度学习的函数相似性检测
  - 总体框架

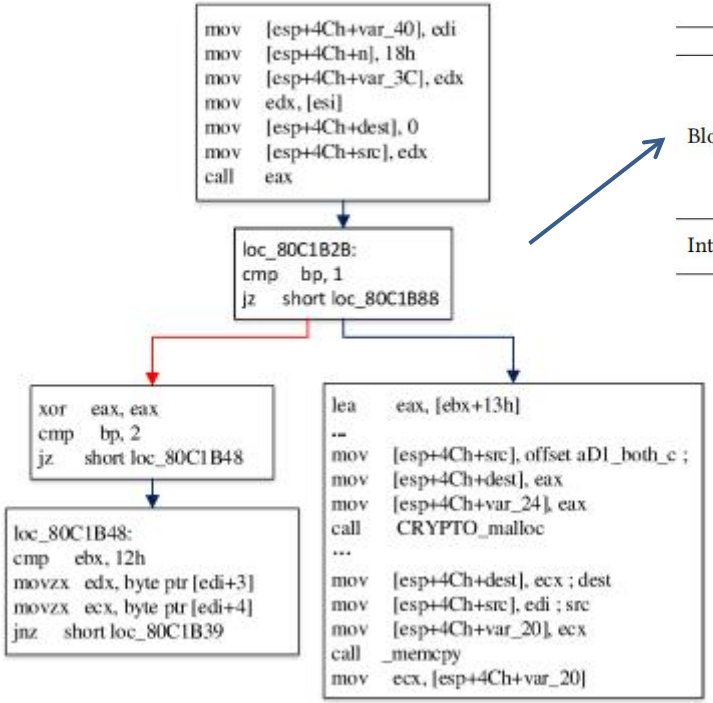


算法原理

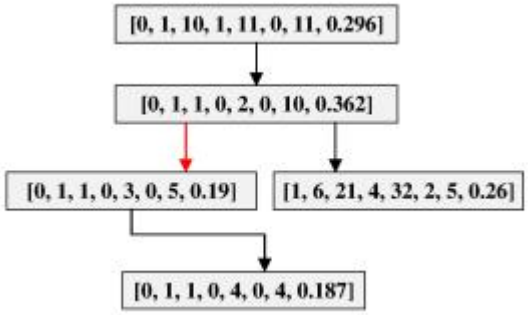
- 基于深度学习的函数相似性检测
  - Embedding Network
    - 目标：将程序嵌入到向量空间中，相似程序的向量相近
    - 方法：首先提取程序的控制流图（CFG），其后使用特征工程将控制流图转换成带节点属性的控制流图（ACFG），最后使用图嵌入网络（Graph Embedding Network）提取ACFG的特征，得到程序的向量表示

算法原理

- 基于深度学习的函数相似性检测
  - Embedding Network
    - 提取ACFG

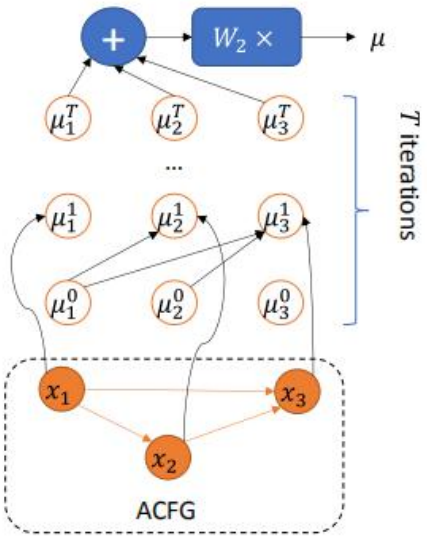


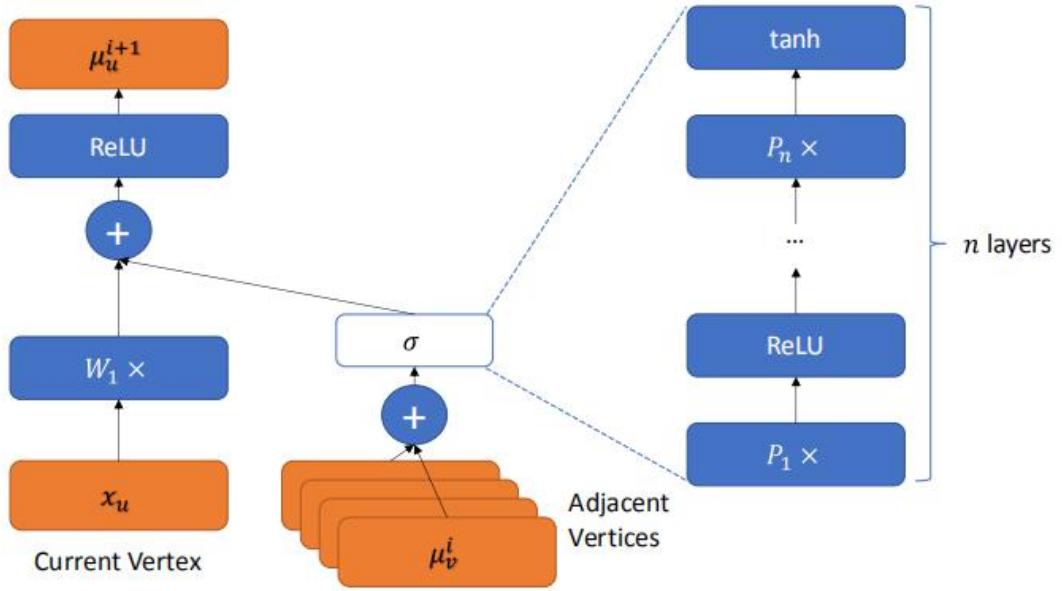
Type	Attribute name
Block-level attributes	String Constants
	Numeric Constants
	No. of Transfer Instructions
	No. of Calls
	No. of Instructions
	No. of Arithmetic Instructions
Inter-block attributes	No. of offspring
	Betweenness



算法原理

- 基于深度学习的函数相似性检测
  - Embedding Network
    - 图嵌入网络





$$\mu_g := A_{v \in \mathcal{V}}(\mu_v)$$

$$\mathcal{F}(x_v, \sum_{u \in \mathcal{N}(v)} \mu_u) = \tanh(W_1 x_v + \sigma(\sum_{u \in \mathcal{N}(v)} \mu_u))$$

优劣分析

- 纵向对比
  - 经典方法：漏报率高、准确率低、效率低、难以适应新样本
  - 深度学习：漏报率低、准确率高、效率高、可以适应新样本

- 横向对比
  - 代码扫描：句子表示学习+TextCNN性能表现最好

参考文献

[1] Xu, Xiaojun, Liu, Chang, Feng, Qian, etc. Neural Network-based Graph Embedding for Cross-Platform Binary Code [J]. 2017.
[2] Yongjun Lee, Hyun Kwon, Sang-Hoon Choi, etc. Instruction2vec: Efficient Preprocessor of Assembly code to Detect Software Weakness with CNN [J]. 2019.
[3] Applying Deep Learning and Vector Representation for Software Vulnerabilities Detection [J]. 2018.

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。