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

基于汇编指令嵌入的漏洞同源性判别

硕士研究生 邢继媛

2021年11月7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

预期收获

- 预期收获
  - 1. 了解漏洞同源性判别任务的基本概念
  - 2. 了解指令嵌入与程序分析的结合方式
  - 3. 理解漏洞同源性判别任务的算法原理

背景简介 震惊全球的“心脏滴血”漏洞

- 2014年，OpenSSL被发现安全漏洞（Heartbleed bug）。
  - SSL：（Secure Socket Layer）安全协议
  - OpenSSL：开源程序；对通信进行SSL加密
  - 影响：泄露用户的登录账号密码、电子邮件甚至是加密秘钥等关键数据

2014-04-08	京东某分站openssl漏洞导致敏感信息泄露及全站随机用户登录(证明可登录)
2014-04-08	豌豆荚运维不当导致服务器敏感信息泄露
2014-04-08	开源中国运维不当导致可随机登录用户并获取服务器敏感信息（成功登陆）
2014-04-08	okcoin运维不当导致可随机登录用户并获取服务器敏感信息
2014-04-08	LastPass运维不当导致敏感信息泄漏漏洞
2014-04-08	网易126邮箱运维不当导致可获取服务器敏感信息
2014-04-08	搜狗通行证服务器运维不当导致信息泄露
2014-04-08	唯品会运维不当导致敏感信息泄漏
2014-04-08	苏宁易购主站运维不当导致可以登录随机用户并且获取服务器敏感信息
2014-04-08	蘑菇街主站运维不当导致可以登录随机用户并且获取服务器敏感信息
2014-04-08	163邮箱运维不当导致可随机登录用户并获取服务器敏感信息
2014-04-08	中国银联运维不当导致可能存在随机登录银联账户并获取服务器敏感信息
2014-04-08	盛大网络运维不当导致敏感信息泄漏
2014-04-08	OWASP主站运维不当导致可获取服务器敏感信息
2014-04-08	360某api运维不当导致敏感信息泄漏
2014-04-08	雅虎主站运维不当导致可以登录随机用户并且获取服务器敏感信息
2014-04-08	微信网页版和公众账号版运维不当导致可随机登录微信用户并获取服务器敏感信息
2014-04-08	比特币中国运维不当导致随机用户明文密码泄漏
2014-04-08	12306新版订票系统运维不当导致可以登录随机用户并且获取服务器敏感信息

背景简介 蜻蜓FM数据造假风波

- 2015年，一名程序猿反编译蜻蜓FM的APP程序并分析（伪）源代码，发现其通过恶意代码造假用户数据。
  - 该APP没有混淆代码，反编译后可以轻易分析出程序逻辑

基本概念

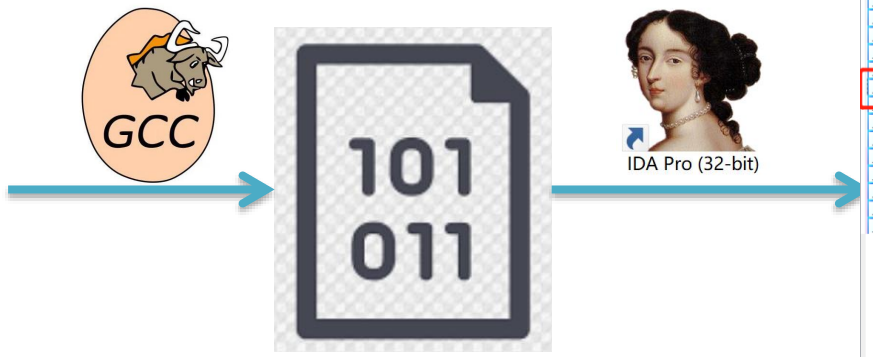
基本概念

- 漏洞同源性判别
  - 定义：判断一个函数是否为漏洞函数的同源函数
- 同源函数
  - 定义：由相同源码编译得到的程序函数
- 软件逆向工程
  - 概念：对程序的结构，流程，算法，代码等进行逆向拆解和分析
  - 应用领域：软件维护，软件破解，漏洞挖掘，恶意代码分析
  - 主要方法：反汇编、反编译、通信数据包分析
  - 主要工具：
    - 静态分析工具：IDA Pro、C32Asm
    - 动态调试工具：OD、DEBUG、x64Dbg

基本概念

- 反汇编
  - 源码->编译->二进制程序->反汇编-> 汇编代码序列/控制流图

```
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;
}
```

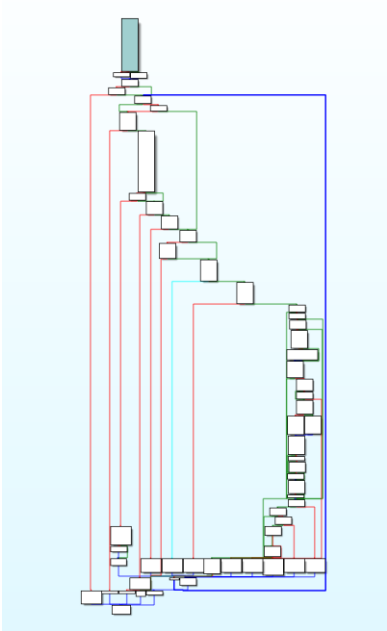



```
.text:0000000000001145 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:0000000000001145                 public main
.text:0000000000001145 main            proc near               ; DATA XREF: start+1D↑o
.text:0000000000001145
.text:0000000000001145 var_4           = dword ptr -4
.text:0000000000001145
.text:0000000000001145 ; __unwind {
.text:0000000000001145                 push    rbp
.text:0000000000001146                 mov     rbp, rsp
.text:0000000000001149                 sub     rsp, 10h
.text:000000000000114D                 lea     rdi, format     ; "Please input a number:"
.text:0000000000001154                 mov     eax, 0
.text:0000000000001159                 call    _printf
.text:000000000000115E                 lea     rax, [rbp+var_4]
.text:0000000000001162                 mov     rsi, rax
.text:0000000000001165                 lea     rdi, aD         ; "%d"
.text:000000000000116C                 mov     eax, 0
.text:0000000000001171                 call    __isoc99_scanf
.text:0000000000001176                 mov     eax, [rbp+var_4]
.text:0000000000001179                 cmp     eax, 0Ah
.text:000000000000117C                 jle     short loc_1189
.text:000000000000117E                 mov     eax, [rbp+var_4]
.text:0000000000001181                 sub     eax, 2
.text:0000000000001184                 mov     [rbp+var_4], eax
.text:0000000000001187                 jmp     short loc_1192
.text:0000000000001189 ; ---------------------------------------------------------------------------
.text:0000000000001189
.text:0000000000001189 loc_1189:                               ; CODE XREF: main+35↑j
.text:0000000000001189                 mov     eax, [rbp+var_4]
.text:000000000000118C                 add     eax, 5
.text:000000000000118F                 mov     [rbp+var_4], eax
.text:0000000000001192
.text:0000000000001192 ; } // starts at 1145
.text:0000000000001192 main            endp
.text:0000000000001192
```

汇编指令序列

基本概念

- 控制流图（CFG）
  - 有向图 G（V，E）
    - 结点：基本块
    - 边：控制流



复杂函数的控制流图包含很多个基本块和边

```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_4= dword ptr -4

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 10h
lea     rdi, format     ; "Please input a number:"
mov     eax, 0
call    _printf
lea     rax, [rbp+var_4]
mov     rsi, rax
lea     rdi, aD         ; "%d"
mov     eax, 0
call    ___isoc99_scanf
mov     eax, [rbp+var_4]
cmp     eax, 0Ah
jle     short loc_1189
```

```
mov     eax, [rbp+var_4]
sub     eax, 2
mov     [rbp+var_4], eax
jmp     short loc_1192
```

```
loc_1189:
mov     eax, [rbp+var_4]
add     eax, 5
mov     [rbp+var_4], eax
```

```
loc_1192:
mov     eax, [rbp+var_4]
mov     esi, eax
lea     rdi, aAD        ; "a = %d"
mov     eax, 0
call    _printf
mov     eax, 0
leave
retn
; } // starts at 1145
main endp
```

基本概念

- 一个函数可以有多少个同源函数?

3（架构）* 3（编译器）* 4（优化选项）* n

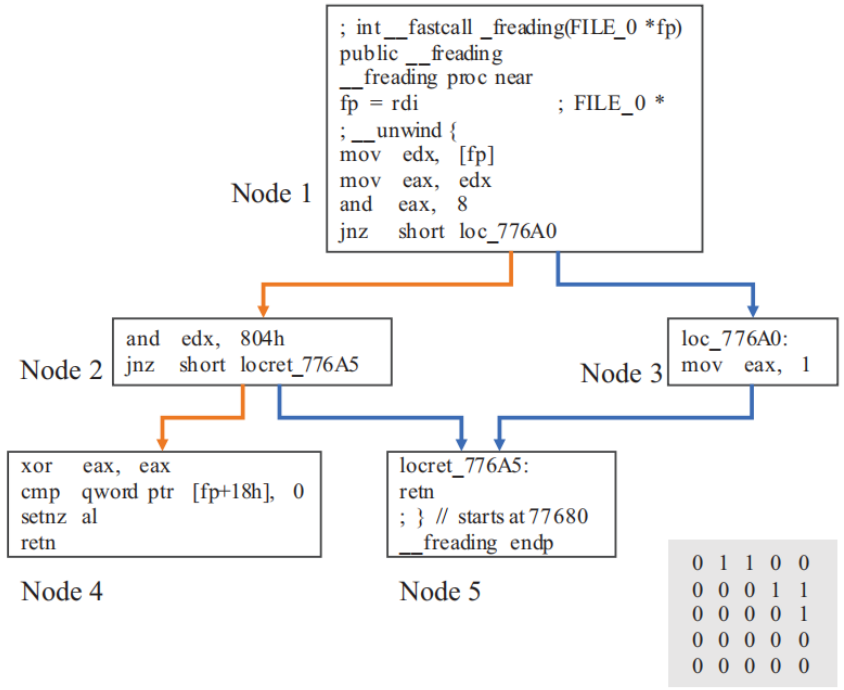
  - 编译过程:
    - CPU指令集架构：X86, ARM, MIPS,……
    - 编译器：GCC, Clang，LLVM ……
    - 优化选项：O0, O1, O2, O3 ……

x86

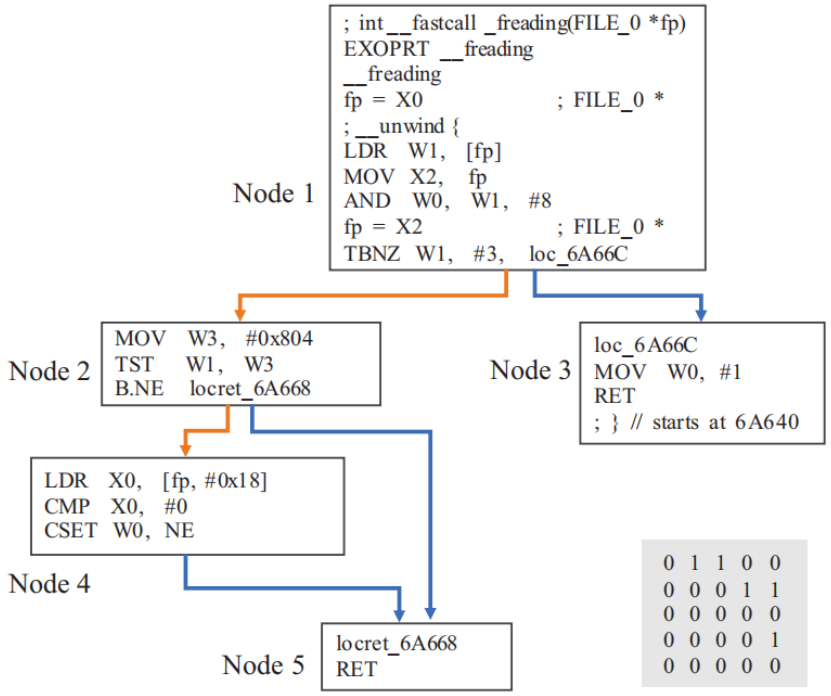
arm

CLANG

- 同源函数判别的难点?
  - 同一函数在不同平台下的控制流程图



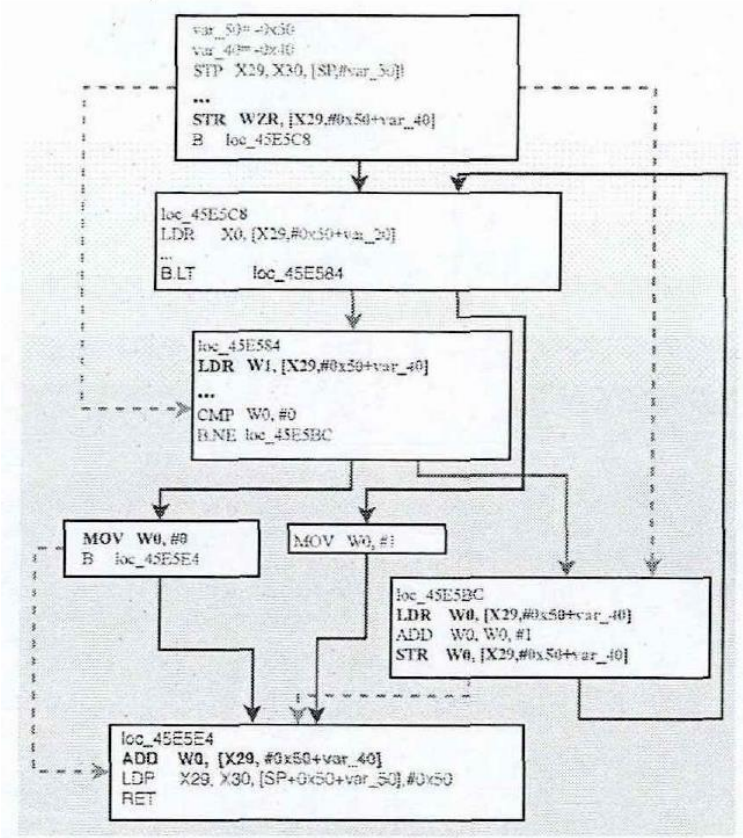
X86-64平台



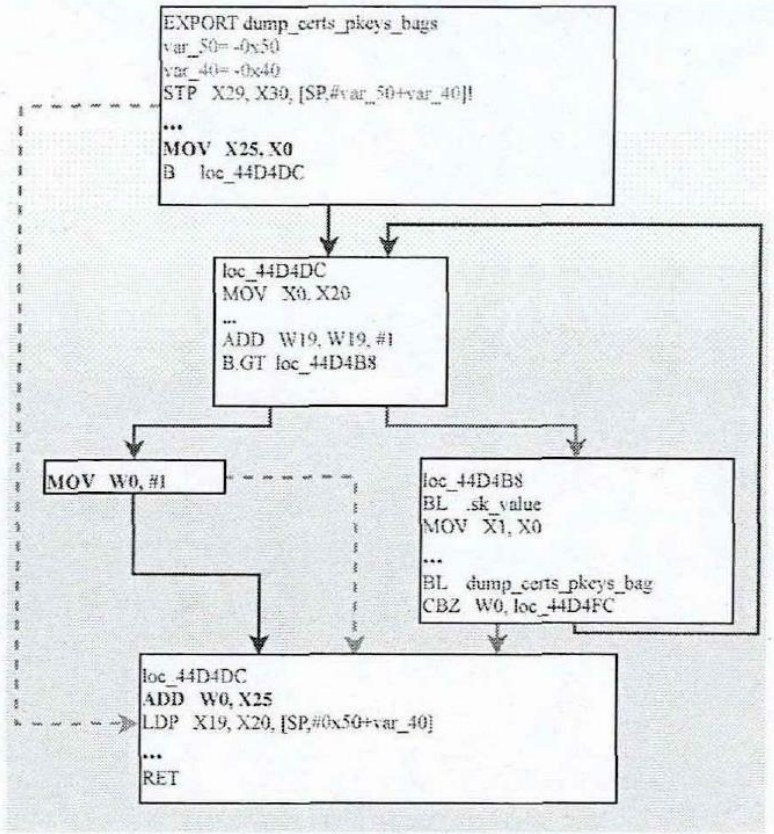
ARM平台

基本概念

- 同源函数判别的难点?
  - 同一函数在不同优化选项下的控制流程图



优化选项：O0



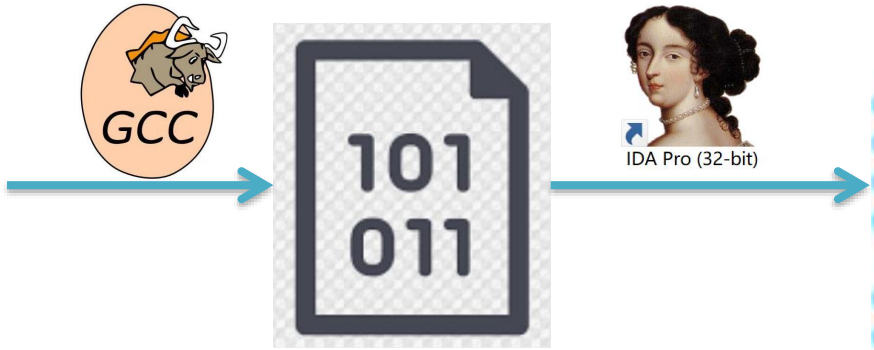
优化选项：O3

基本概念

- 反编译 （无混淆）

```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;
}
```

源码



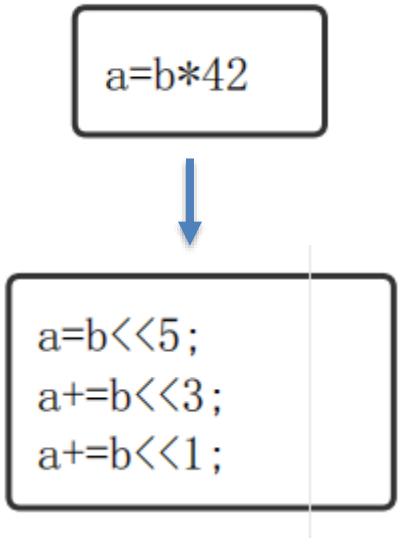

```c
Pseudocode-A
1 int __cdecl main(int argc, const char **argv, cc
2 {
3   unsigned int v4; // [rsp+Ch] [rbp-4h]
4
5   printf("Please input a number:", argv, envp);
6   __isoc99_scanf("%d", &v4);
7   if ( (signed int)v4 <= 10 )
8     v4 += 5;
9   else
10    v4 -= 2;
11  printf("a = %d", v4);
12  return 0;
13}
```

反编译工具IDA生成的伪源码

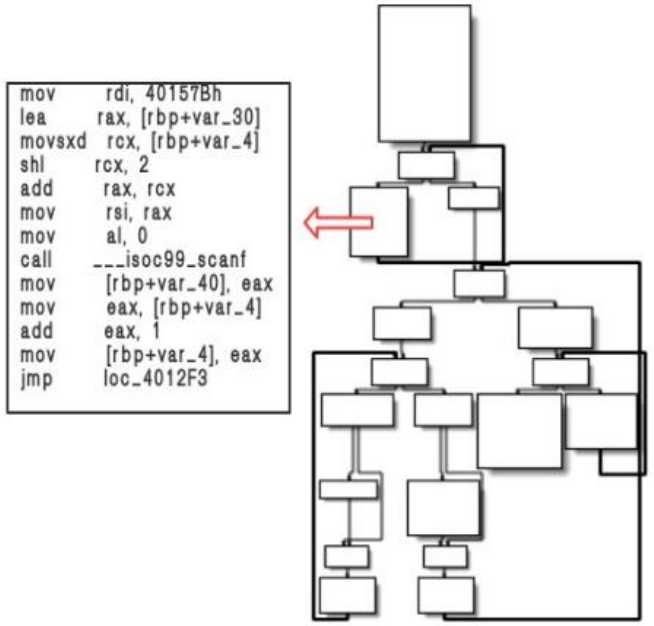
基本概念

- 代码混淆
  - 目标：将计算机程序的源代码或机器代码，转换成功能等价，但是难于阅读和理解的形式
  - 阶段：编译阶段
  - 好处：增加反编译难度；提高安全性；
  - 混淆策略：
    - 布局混淆
    - 数据流混淆
    - 控制流混淆
    - 预防混淆

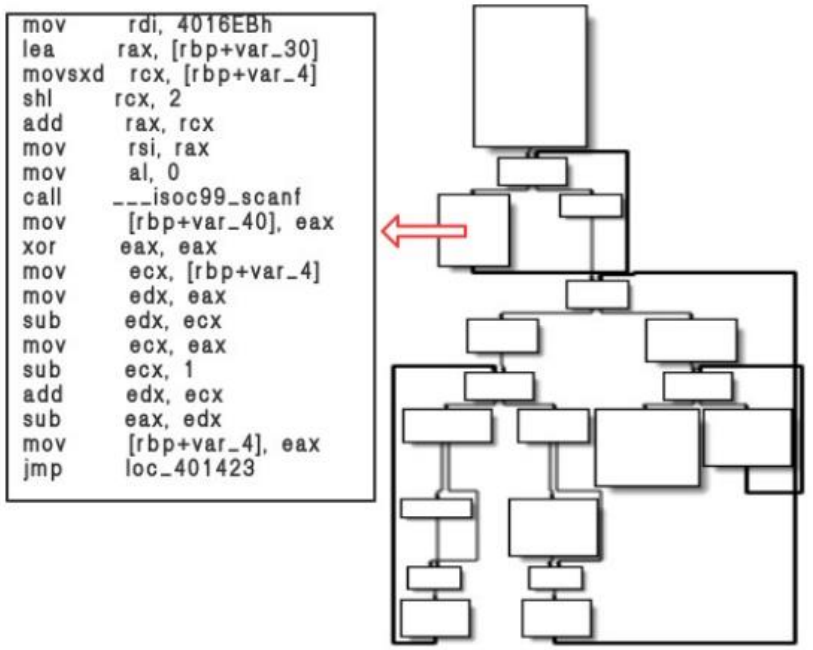
- C语言混淆工具：OLLVM
  - 混淆方式1：指令替换（ SUB ）
    - 思路：恒等运算
    - 结果：控制流图结构没有改变，增加算术指令、逻辑指令和常量数量



恒等运算



混淆前

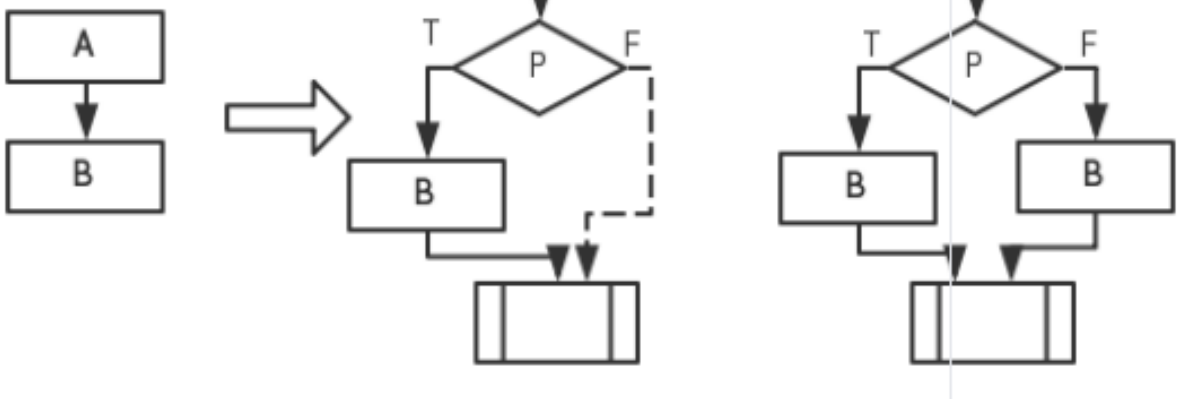


混淆后

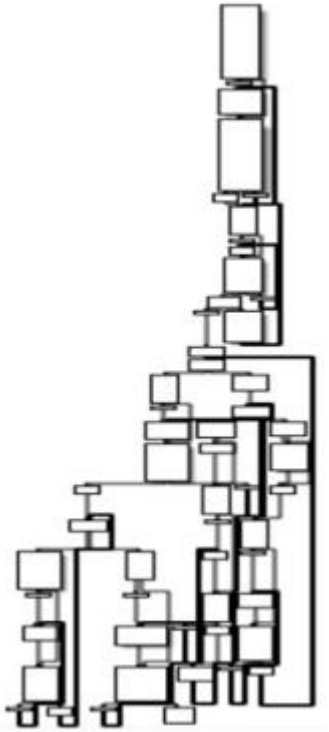
基本概念

- 混淆方式2：虚假控制流 （BCF）
  - 思路：不透明谓词（值已知，但难以破解的表达式 ）

    （例：((x+1)*x)%2==0 ）
  - 结果：CFG中控制流分支错综复杂，节点数增加，垃圾指令增多



混淆前



混淆后

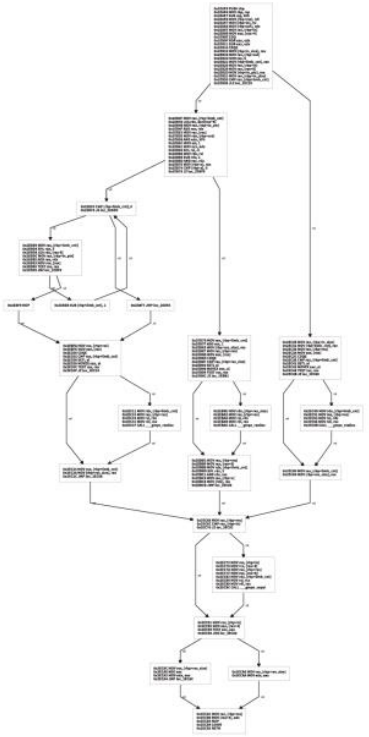
基本概念

– 混淆方式3：控制流平展（FLA）

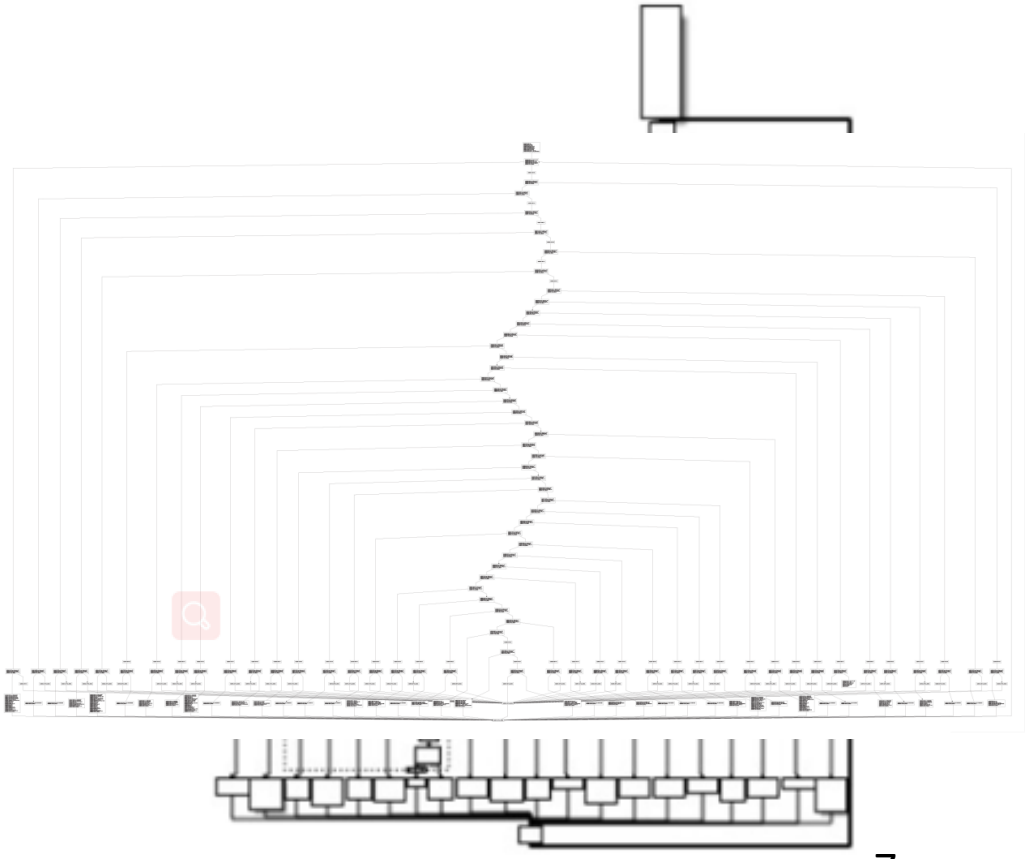
- 思路：压扁控制流（将条件转移和嵌套循环的控制结构平展）
- 结果：CFG失去原本的结构；节点、垃圾指令、伪分支增多

```
int modexp(int y, int x[], int w, int n)
{
    int R, L;
    int k = 0;
    int s = 1;
    while(k < w)
    {
      if(x[k] == 1)
          R = s * y % n;
      else
          R = s;
      s = R * R % n;
      L = R;
      k++;
    }
    return L;
}
```

```
int modexp(int y, int x[], int w, int n)
{
    int R, L, k, s;
    int next = 0;
    for(;;)
    {
      switch(next)
      {
        case 0 : k = 0; s = 1; next = 1; break;
        case 1 : if(k < w) next = 2; else next = 6; break;
        case 2 : if(x[k] == 1) next = 3; else next = 4; break;
        case 3 : R = s * y % n; next = 5; break;
        case 4 : R = s; next = 5; break;
        case 5 : s = R * R % n; L = R; k++; next = 1; break;
        case 6 : return L;
      }
    }
}
```



混淆前



混淆后

算法原理

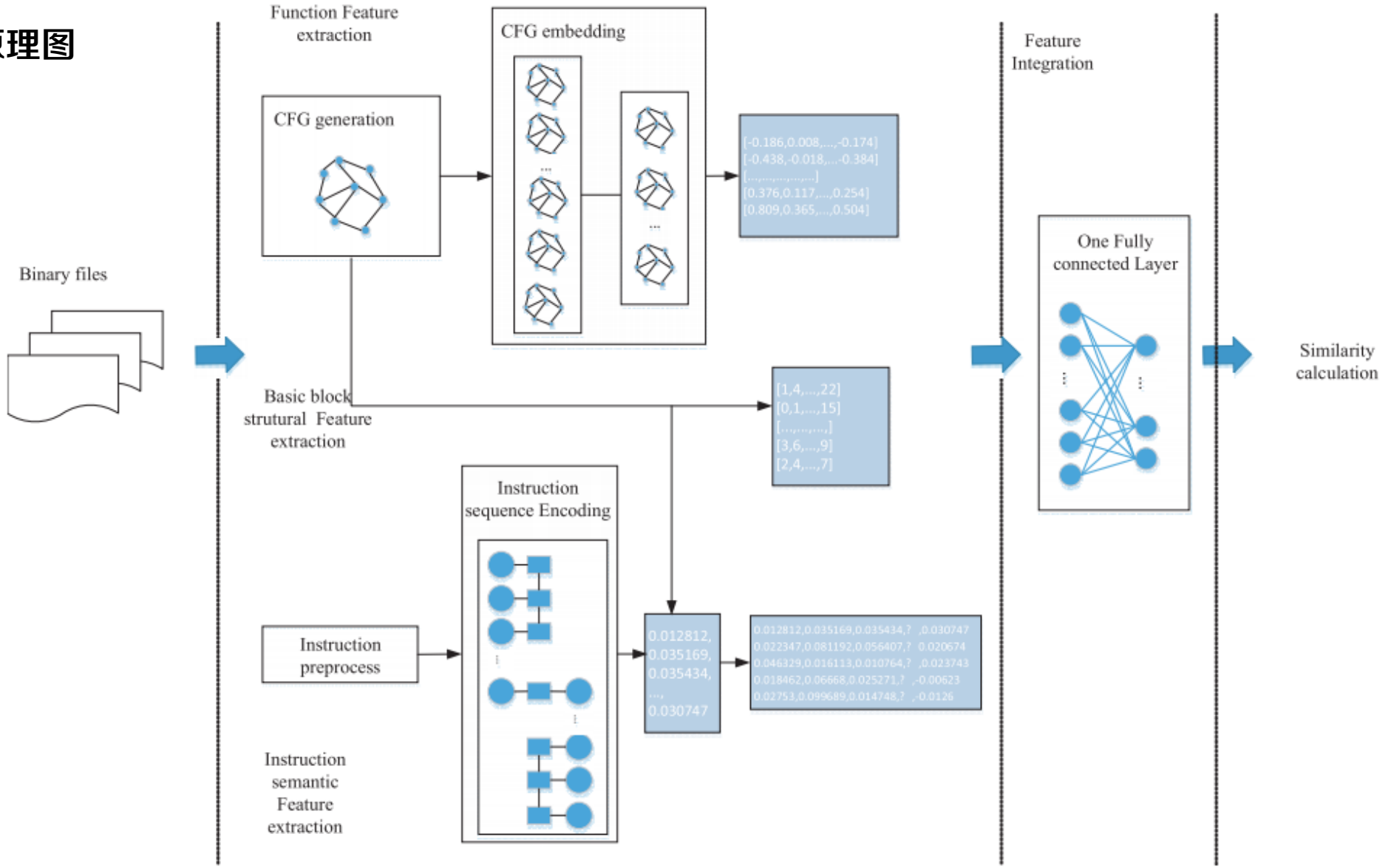
算法原理-BEDetector

T	二进制程序函数同源性判别
I	两个二进制程序函数
P	1. 语义特征提取 2. 结构特征提取 3. 特征融合 4. 距离计算
O	二分类；同源/非同源

P	充分提取函数的语义信息与结构（控制逻辑）信息
C	二进制程序函数未混淆
D	不同类型特征进行融合
L	2021 IEEE Access

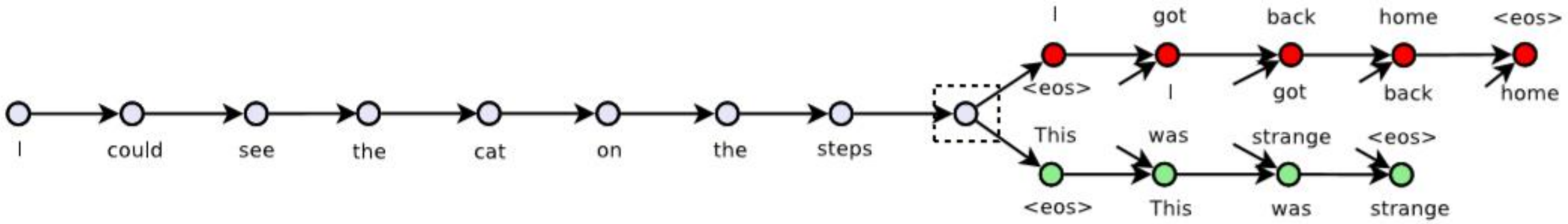
算法原理

算法原理图


Binary files
Function Feature extraction
CFG generation
CFG embedding
[-0.186,0.008,...,-0.174]
[-0.438,-0.018,...-0.384]
[0.376,0.117,...,0.254]
[0.809,0.365,...,0.504]
Basic block strutural Feature extraction
[1,4,...,22]
[0,1,...,15]
[3,6,...,9]
[2,4,...,7]
Instruction sequence Encoding
Instruction preprocess
Instruction semantic Feature extraction
0.012812, 0.035169, 0.035434, ..., 0.030747
0.012812,0.035169,0.035434,? ,0.030747
0.022347,0.081192,0.056407,? 0.020674
0.046329,0.016113,0.010764,? ,0.023743
0.018462,0.06668,0.025271,? ,-0.00623
0.02753,0.099689,0.014748,? ,-0.0126
Feature Integration
One Fully connected Layer
Similarity calculation

算法原理-语义特征提取

- 指令粒度
  - 预备知识：Skip-thoughts 模型
    - 无监督句子表示模型
    - 一个编码器Encoder，两个Decoder：分别解码出当前句子的上一句和下一句
    - 模型的Encoder部分作为句子特征提取器，生成句子的向量表示



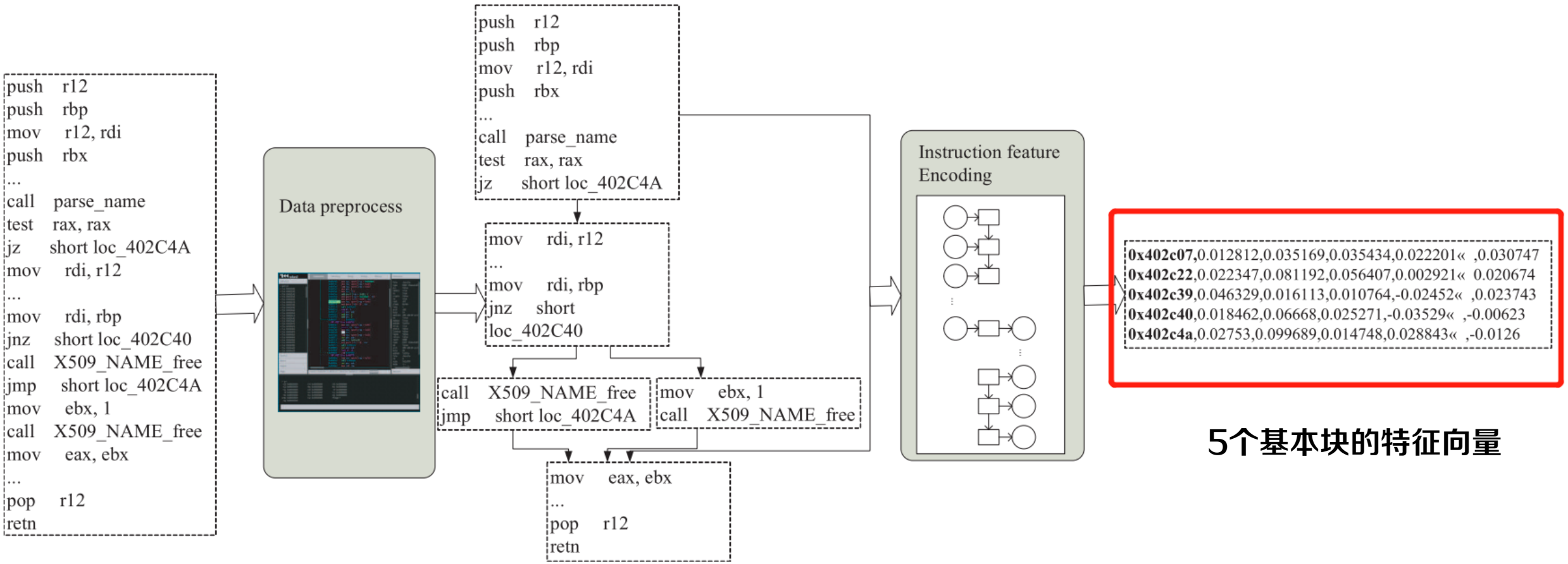
算法原理-语义特征提取

- 指令粒度
  - 单词 -> 句子 ->段落
  - 操作数/码->汇编指令->基本块
- 指令预处理：
  - OOV问题：out-of-vocabulary
  - 操作数类型：寄存器/立即数
- 语义特征提取：
  - 输入：基本块中预处理后的指令
  - 训练：Encoder+ Decoder
  - 输出：使用Encoder生成基本块的向量表示

mov	eax , 0x01
操作码	目的操作数 源操作数

指令预处理方法	
立即数值	替换值
0-0x100	IMM1
0x101-0x200	IMM2
>0x200	IMM

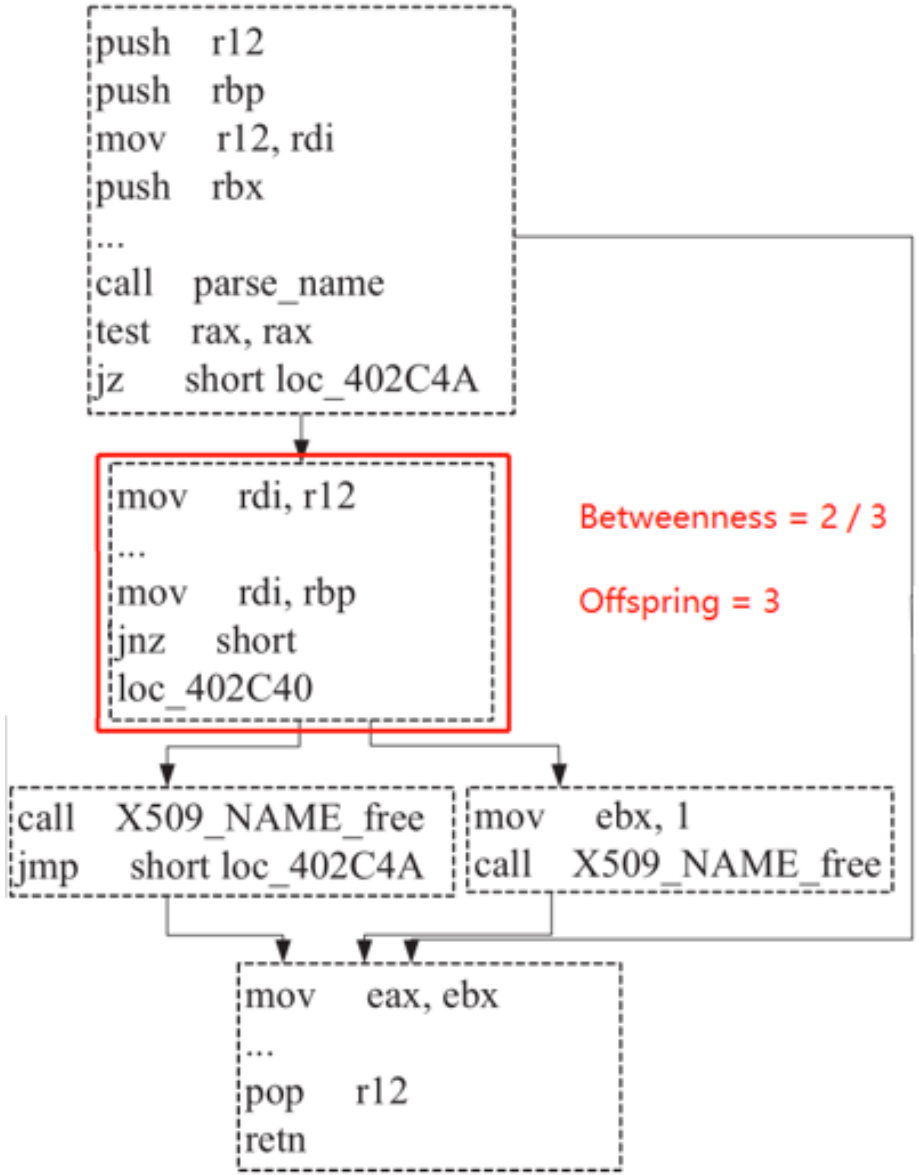
算法原理—语义特征提取



5个基本块的特征向量

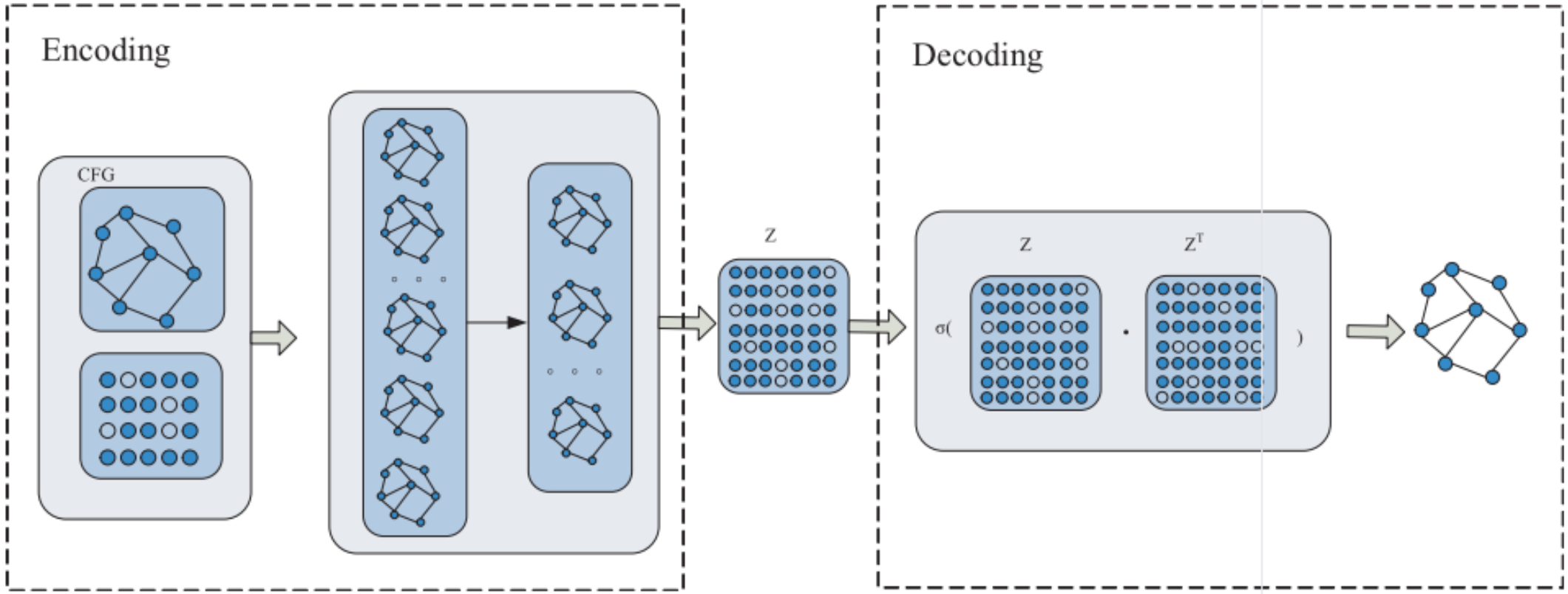
算法原理—结构特征提取

- 结构特征提取
  - 基本块粒度：
    - Betweenness：
      - 通过该基本块的路径数/路径总数；
      - 表示基本块对于控制流图的结构影响；
    - Offspring：
      - 该基本块连接的基本块数目；
      - 表示基本块对于控制流图的结构影响；



算法原理—结构特征提取

- 结构特征提取
  - 函数粒度：
    - 预备知识：图自编码器（GAE）

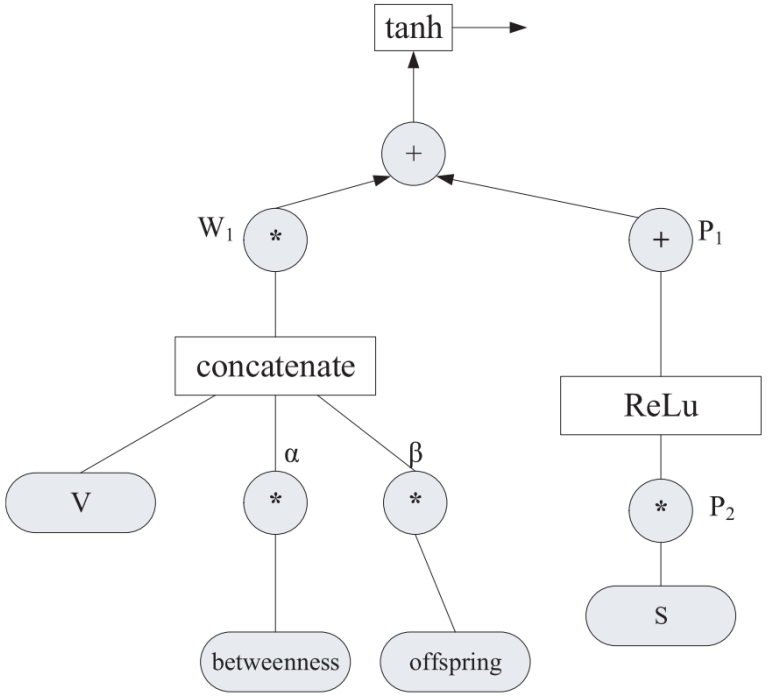


算法原理—特征融合

- 特征融合
  - 指令粒度：基本块中的指令
  - 基本块粒度： Offspring、 Betweenness
  - 函数粒度：控制流图的拓扑信息

语义特征

结构特征



Algorithm 1 Feature Integration in Basic Block Granularity

/*V_i is the feature vector of basic block B_i in function F*/
/*T_i and P_i is the betweenness and offspring of B_i*/
/*N is the total number of basic blocks in function F.*/

Begin:

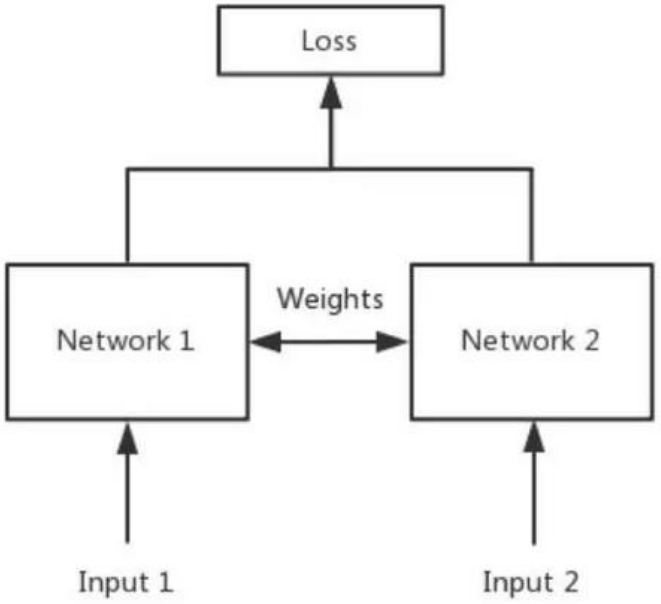
1: For vector V_i, let $v_i^1, \ldots, v_i^j, \ldots, v_i^M$ be the elements of V_i where M is the vector dimension.
2: Get the betweenness and offspring value T_i and P_i and assign them weight value α, β.
3: Concatenate value of V_i, αT_i and weighted value βP_i of betweenness and offspring.
4: Define $V_i' = (v_i^1, \ldots, v_i^M, \alpha T_i, \beta P_i)$.
5: $D=(V_1', V_2', \ldots, V_N')^T$.
6: return D

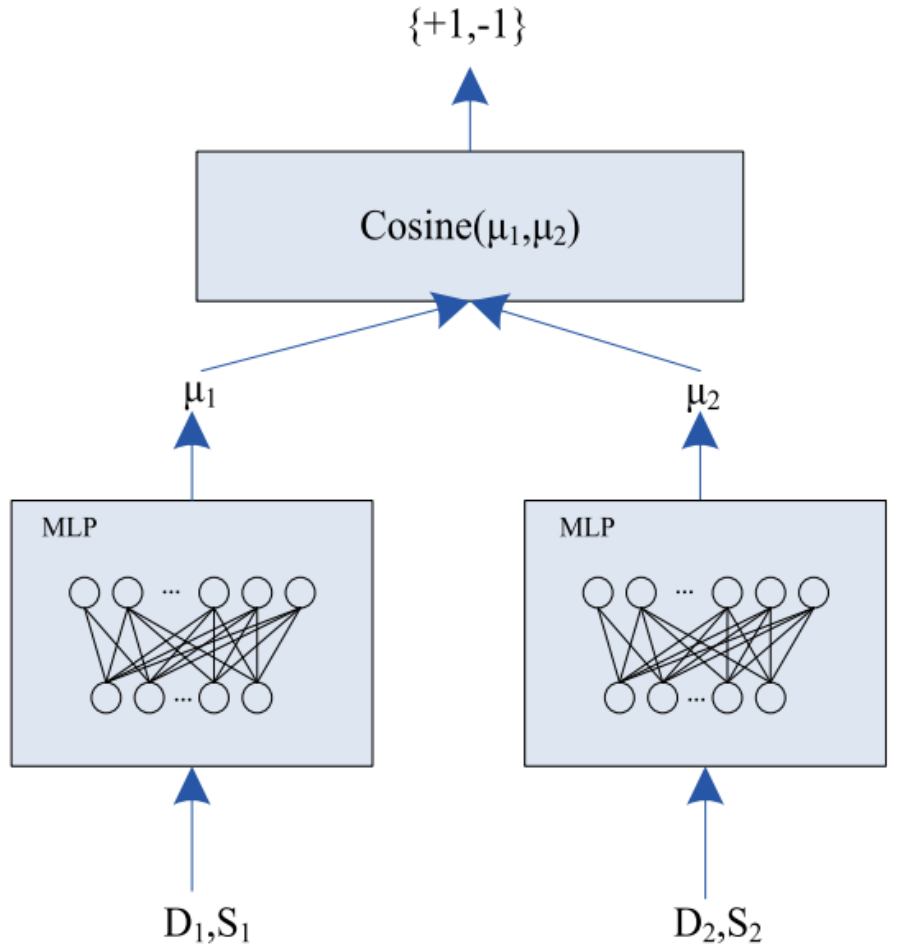
End

$$M = tanh(DW_1 + ReLu(SP_2)P_1)$$

算法原理

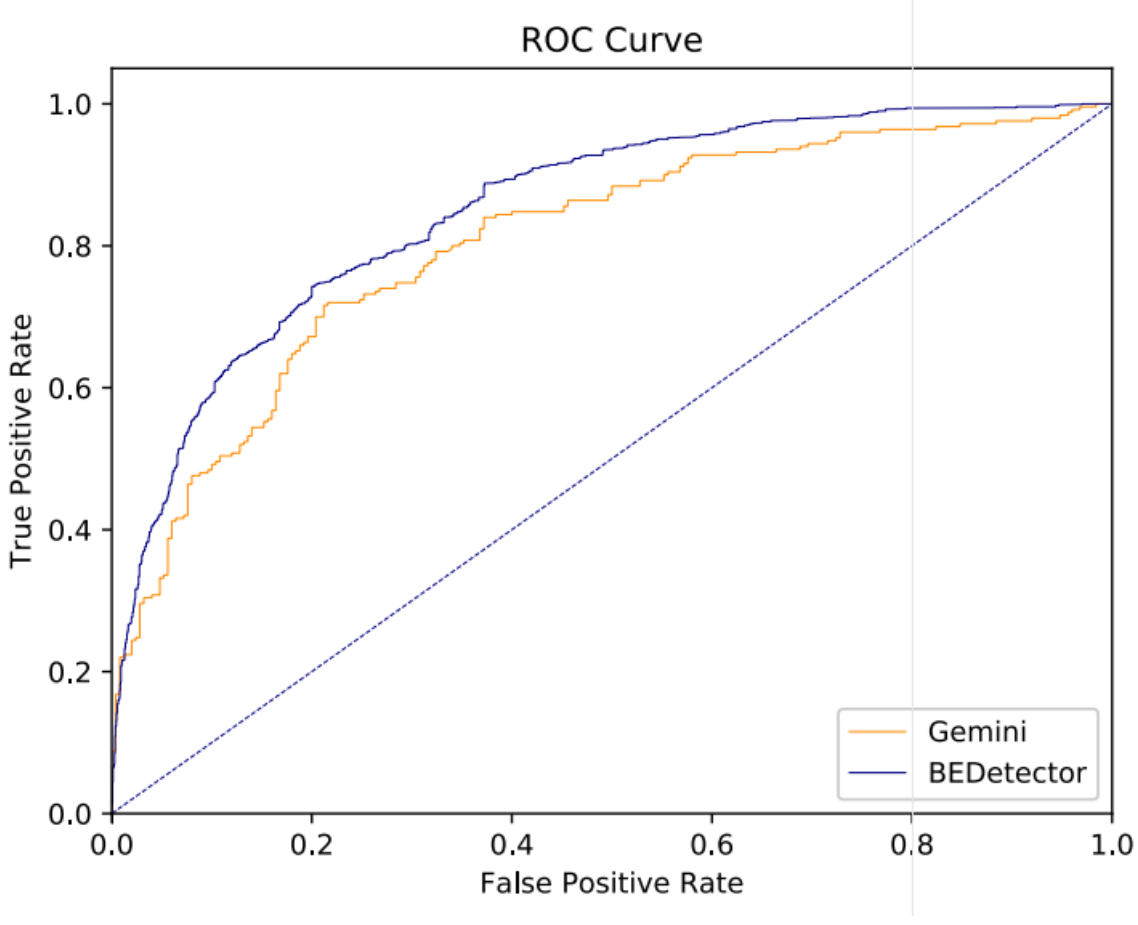
- 模型训练：孪生神经网络
  - 共享权值
  - 应用场景：两个输入“类似”的情况
- 训练目标：缩小同源函数向量表示之间的距离，
  增大非同源函数向量表示间距离。





实验结果

- 同源函数判别
  - 程序： OpenSSL BusyBox
  - 架构：x86-32, x86-64, MIPS32, MIPS64, ARM32, ARM64



实验结果

- 同源漏洞搜索
  - 固件映像数据集
  - 400个固件中是否存在以下三个漏洞函数
    - ssl3_get_key_exchange (CVE-2015-0204)
    - ssl3_get_new_se-ssion_ticket (CVE-2015-1791)
    - udhcp_get_option (CVE-2018-20679)

语义特征

结构特征

Model	MRR/top-1-O0	MRR/top-1-O1	MRR/top-1-O2	MRR/top-1-O3
Node2vec [34]	0.4129/0.3984	0.4018/0.3805	0.3407/0.3128	0.2715/0.2527
DeepWalk [35]	0.2539/0.2152	0.2304/0.2080	0.1596/0.1345	0.1499/0.1187
GCN [36]	0.4241/0.4003	0.4090/0.3929	0.2720/0.2510	0.2598/0.2399
Structure2vec [18]	0.6359/0.5848	0.6290/0.5920	0.5530/0.5030	0.5428/0.4835
Word2vec+GAE	0.7526/0.7028	0.7335/0.6985	0.6892/0.6425	0.6638/0.6358
Our model	0.7904/0.7215	0.7729/0.7101	0.7452/0.7039	0.6829/0.6521

- 语义特征在同源性判别中起重要左右
- 句子嵌入模型优于词嵌入模型
- 在真实固件映像中进行同源漏洞搜索仍存在很大提升空间

算法原理

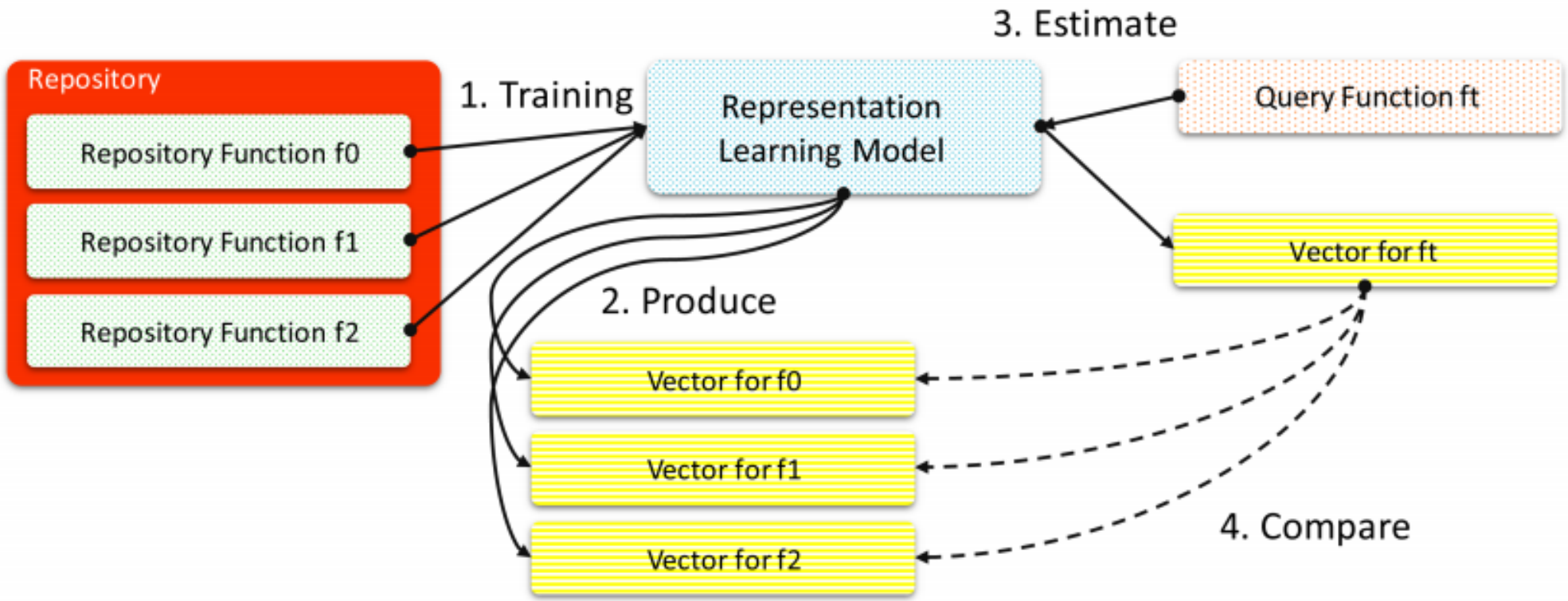
算法原理-Asm2vec

T	同源函数搜索
I	目标函数
P	1. 基于汇编函数存储库，训练汇编函数表示学习模型 2. 模型为存储库中每个函数生成向量表示 3. 模型为目标函数生成向量表示 4. 分别计算库函数与目标函数向量表示的余弦相似度，相似度最高的k个候选函数作为结果
O	存储库中目标函数的同源函数

P	现有方法没有重点突出函数的关键部分
C	所有函数为同一指令架构
D	代码混淆导致控制流图发生巨大变化
L	2019 S&P

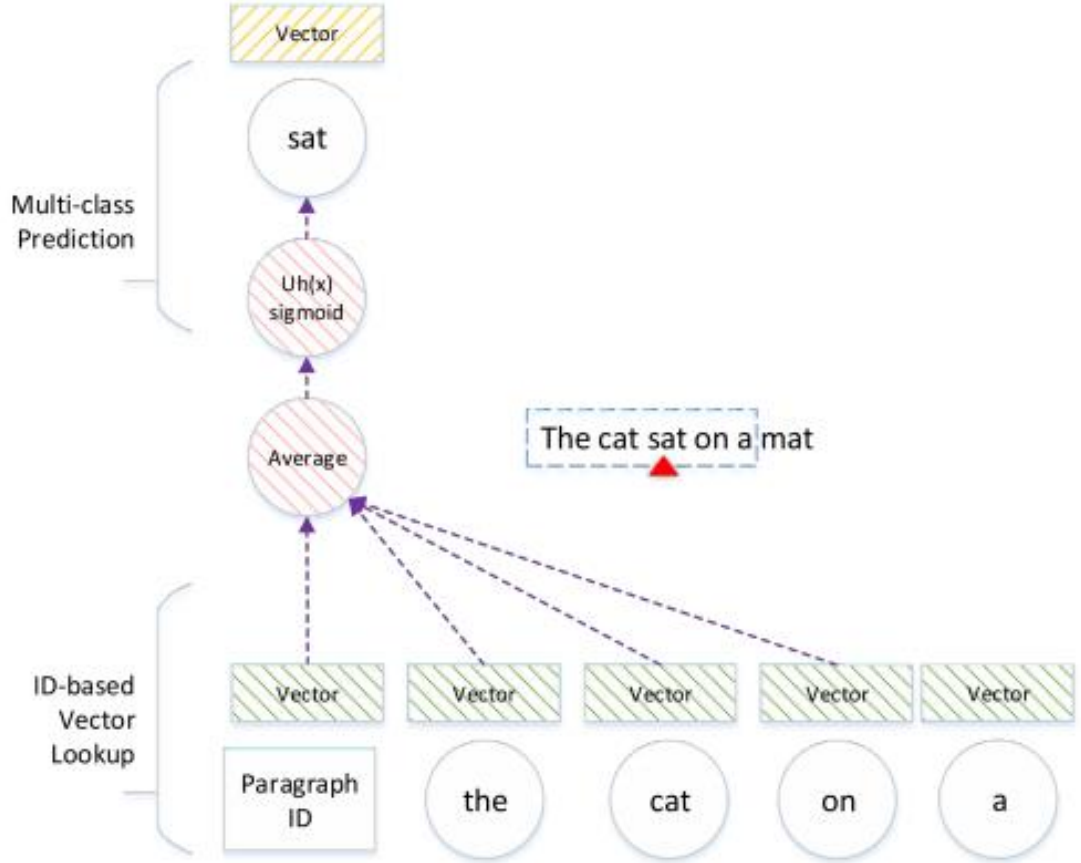
算法原理

- 工作流程图


3. Estimate
Repository
Repository Function f0
Repository Function f1
Repository Function f2
1. Training
Representation Learning Model
Query Function ft
Vector for ft
2. Produce
Vector for f0
Vector for f1
Vector for f2
4. Compare

算法原理-汇编函数表示学习

- 预备知识：PV-DM模型
  - 联合学习每个单词和每个段落的向量表示
  - 文档->段落->句子->单词



$$\sum_{p}^{T}\sum_{s}^{p}\sum_{t=k}^{|s|-k} log\ \mathbf{P}(w_t|p, w_{t-k}, \ldots, w_{t+k})$$

算法原理-汇编函数表示学习

- 汇编指令与文本文档的差异
  - 操作码、操作数
  - 控制流

喜马拉雅FM曝光给媒体的内容显示，其程序员通过反编译蜻蜓FM的手机应用程序，获得其程序源代码，发现三大问题。

首先，蜻蜓FM会定时地执行“普罗米修斯神逻辑”的程序，其作用是在用户不知情的情况下偷偷运行蜻蜓FM的程序，达到增加每日活跃用户数的目的；

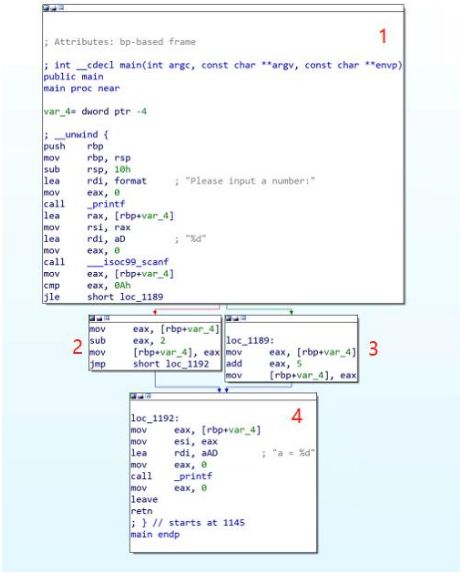
其次，蜻蜓FM代码中代号为“宙斯”的后台程序，在手机后台打开一个隐藏的浏览器，模拟用户访问广告链接，最终导致广告的展示量和点击量大幅虚假提升，并且导致用户手机耗电异常；

最后，蜻蜓FM将这些虚假的月活跃用户数报给第三方数据监测公司，以达到欺瞒广告主、投资人的目的。

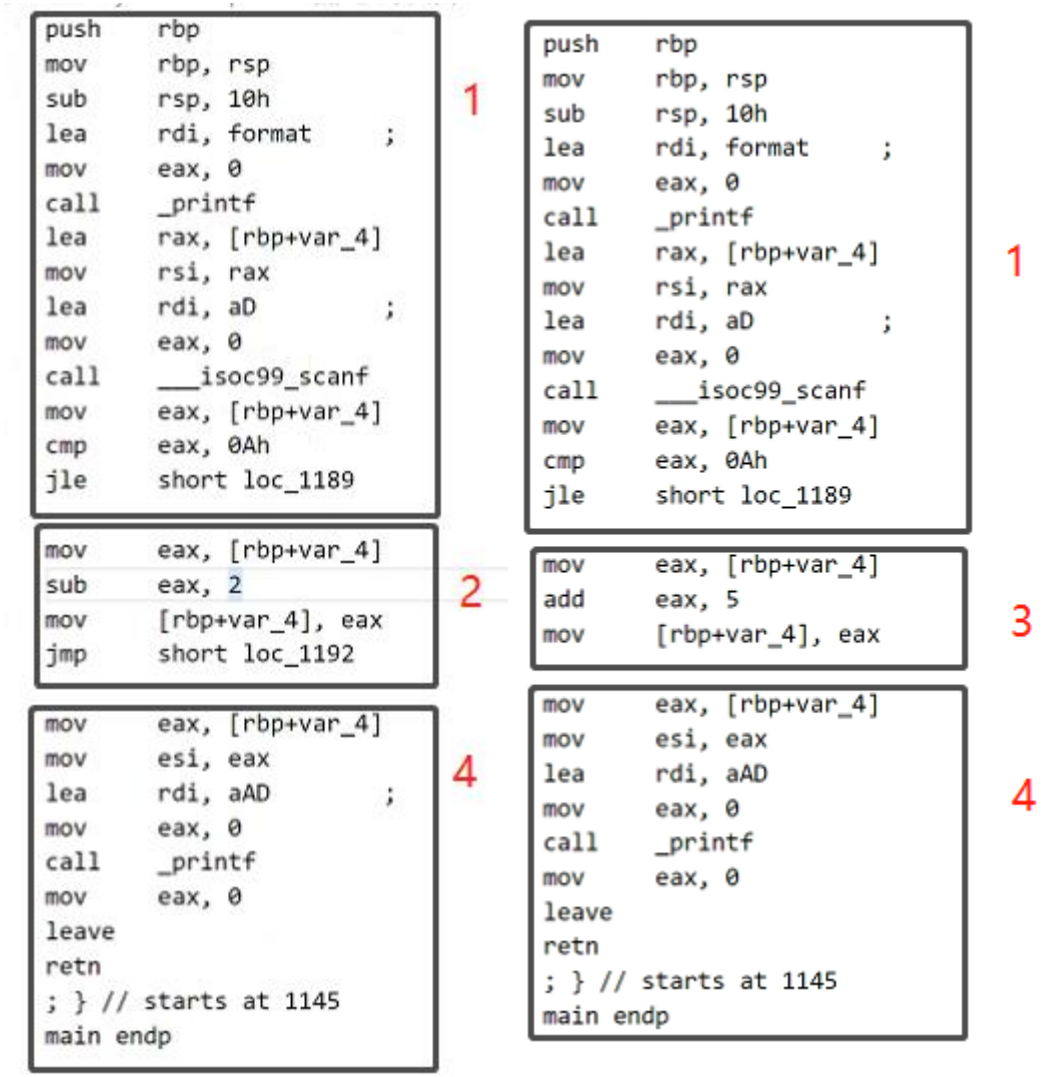
```
{
        push    rbp
        mov     rbp, rsp
        sub     rsp, 10h
        lea     rdi, format     ;
        mov     eax, 0
        call    _printf
        lea     rax, [rbp+var_4]
        mov     rsi, rax
        lea     rdi, aD         ;
        mov     eax, 0
        call    ___isoc99_scanf
        mov     eax, [rbp+var_4]
        cmp     eax, 0Ah
        jle     short loc_1189
        mov     eax, [rbp+var_4]
        sub     eax, 2
        mov     [rbp+var_4], eax
        jmp     short loc_1192
-------------------------------

                                ;
        mov     eax, [rbp+var_4]
        add     eax, 5
        mov     [rbp+var_4], eax

                                ;
        mov     eax, [rbp+var_4]
        mov     esi, eax
        lea     rdi, aAD        ;
        mov     eax, 0
        call    _printf
        mov     eax, 0
        leave
        retn
rts at 1145
        endp
```

算法原理-汇编函数表示学习

- 将汇编函数建模为多个指令序列
  - 1. 边缘覆盖
    - 每个指令序列表示函数潜在的执行路径
    - 即使控制流图中的基本块被分割或合并，由于函数的执行逻辑不变，仍然可以产生相似的序列



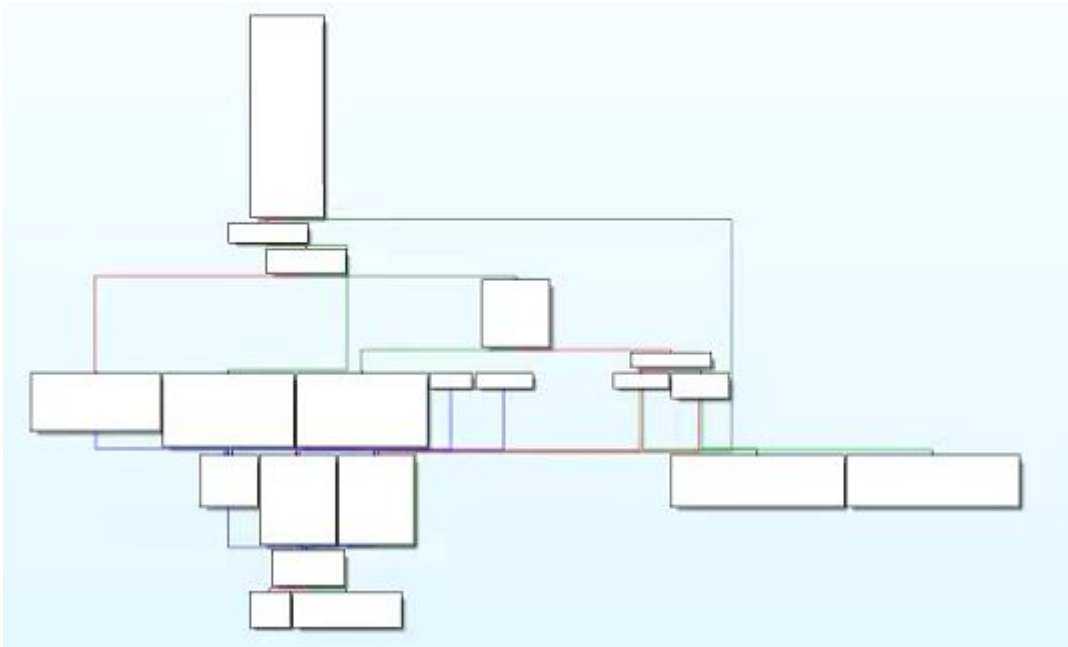
函数



序列1　　序列2

算法原理-汇编函数表示学习

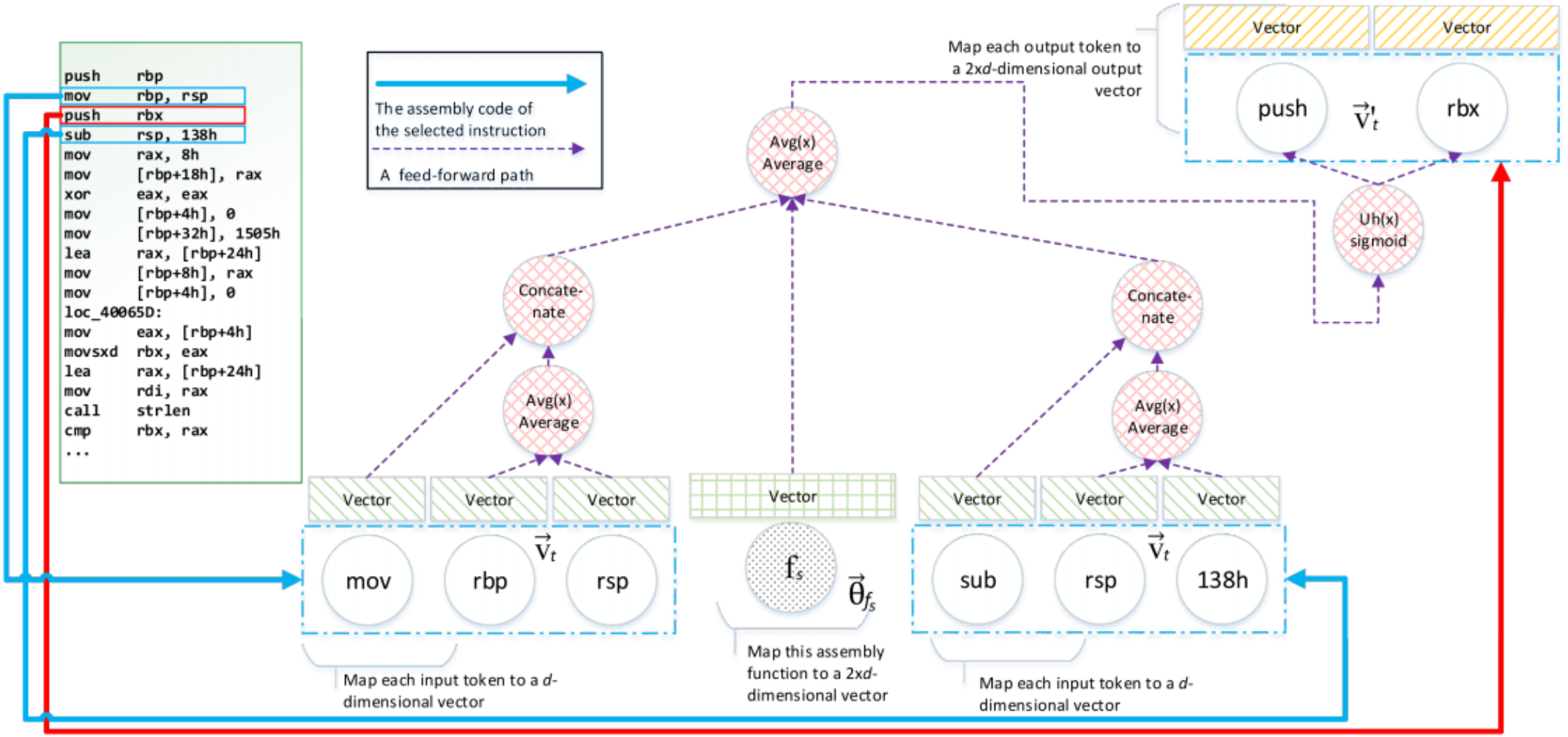
- 将汇编函数建模为多个指令序列
  - 2. 随机游走
    - Dominator：如果基本块B必须通过基本块A才能到达其他块，则基本块A控制基本块B
    - 使具有控制属性的基本块优先级更高



算法原理-Asm2Vec模型

- 文档-> 段 落 -> 句 子 ->单词
- 函数->指令序列->汇编指令->操作数/码

$$\mathrm{RP} \sum_{f_s} \sum_{seq_i} \mathcal{S}(f_s) \sum_{in_j} \mathcal{I}(seq_i) \sum_{t_c} \mathcal{T}(in_j) \; log\, \mathbf{P}(t_c | f_s, in_{j-1}, in_{j+1})$$



实验结果

- 代码混淆后的同源函数搜索

	O-LLVM - Bogus Control Flow Graph (BCF)					O-LLVM - Instruction Substitution (SUB)				
Baselines	Libgmp	ImageMagick	LibTomCrypt	OpenSSL	Avg.	Libgmp	ImageMagick	LibTomCrypt	OpenSSL	Avg.
Composite	.226	.224	.312	.246	.252	.620	.675	.600	.766	.665
Constant	.130	.592	.412	.318	.363	.173	.622	.492	.360	.412
Graphlet	.003	.005	.033	.007	.012	.198	.158	.411	.308	.269
Graphlet-C	.112	.118	.165	.124	.130	.626	.572	.539	.585	.581
Graphlet-E	.026	.011	.050	.014	.025	.454	.216	.286	.271	.307
MixedGram	.220	.234	.375	.303	.283	.585	.642	.563	.743	.633
MixedGraph	.011	.007	.049	.014	.020	.356	.325	.495	.488	.416
n-gram	.134	.134	.295	.195	.190	.466	.516	.513	.670	.541
n-perm	.233	.224	.374	.274	.276	.557	.624	.558	.736	.619
FuncSimSearch	.109	.022	.029	.027	.047	.685	.442	.699	.330	.539
PV(DM/DBOW)	.784	.870	.842	.768	.816	.935	.968	.964	.958	.956
Asm2Vec *	**.802**	**.920**	**.933**	**.883**	**.885**	**.940**	**.960**	**.981**	**.961**	**.961**
	O-LLVM - Control Flow Flattening (FLA)					O-LLVM - SUB+FLA+BCF				
Baselines	Libgmp	ImageMagick	LibTomCrypt	OpenSSL	Avg.	Libgmp	ImageMagick	LibTomCrypt	OpenSSL	Avg.
Composite	.138	.129	.052	.027	.086	.219	.226	.015	.009	.117
Constant	.105	.480	.215	.209	.252	.137	.591	.173	.159	.265
Graphlet	.000	.002	.000	.000	.000	.000	.005	.000	.000	.001
Graphlet-C	.003	.008	.000	.001	.003	.107	.124	.000	.000	.058
Graphlet-E	.001	.002	.000	.000	.001	.020	.012	.000	.000	.008
MixedGram	.148	.143	.075	.036	.101	.221	.234	.018	.010	.121
MixedGraph	.003	.003	.000	.000	.002	.006	.010	.000	.000	.004
n-gram	.095	.093	.059	.030	.069	.154	.144	.013	.007	.079
n-perm	.133	.126	.055	.033	.087	.224	.222	.018	.008	.118
FuncSimSearch	.095	.001	.004	.008	.027	.110	.025	.003	.008	.037
PV(DM/DBOW)	**.852**	**.938**	.786	.763	.835	.780	.873	.639	.595	.722
Asm2Vec *	.772	.920	**.890**	**.795**	**.844**	**.854**	**.880**	**.830**	**.690**	**.814**

- 指令嵌入模型对于结果影响不大
- 模型可以很好的捕捉汇编指令的语义信息
- 模型可以从噪声中识别出函数的关键部分

实验结果

- 同源漏洞搜索
  - 任务：从库（3015个汇编函数）中分别搜索已知漏洞的同源函数
  - 结果：
    - 搜索排名中，前15个候选者均为对应的同源函数
    - 精度及召回率均为1

		ESH [18]			Asm2Vec		
Vulnerability	CVE	FP	ROC	CROC	FP	ROC	CROC
Heartbleed	2014-0160	0	1	1	0	1	1
Shellshock	2014-6271	3	0.999	0.996	0	1	1
Venom	2015-3456	0	1	1	0	1	1
Clobberin' Time	2014-9295	19	0.993	0.956	0	1	1
Shellshock #2	2014-7169	0	1	1	0	1	1
ws-snmp	2011-0444	1	1	0.997	0	1	1
wget	2014-4877	0	1	1	0	1	1
ffmpeg	2015-6826	0	1	1	0	1	1

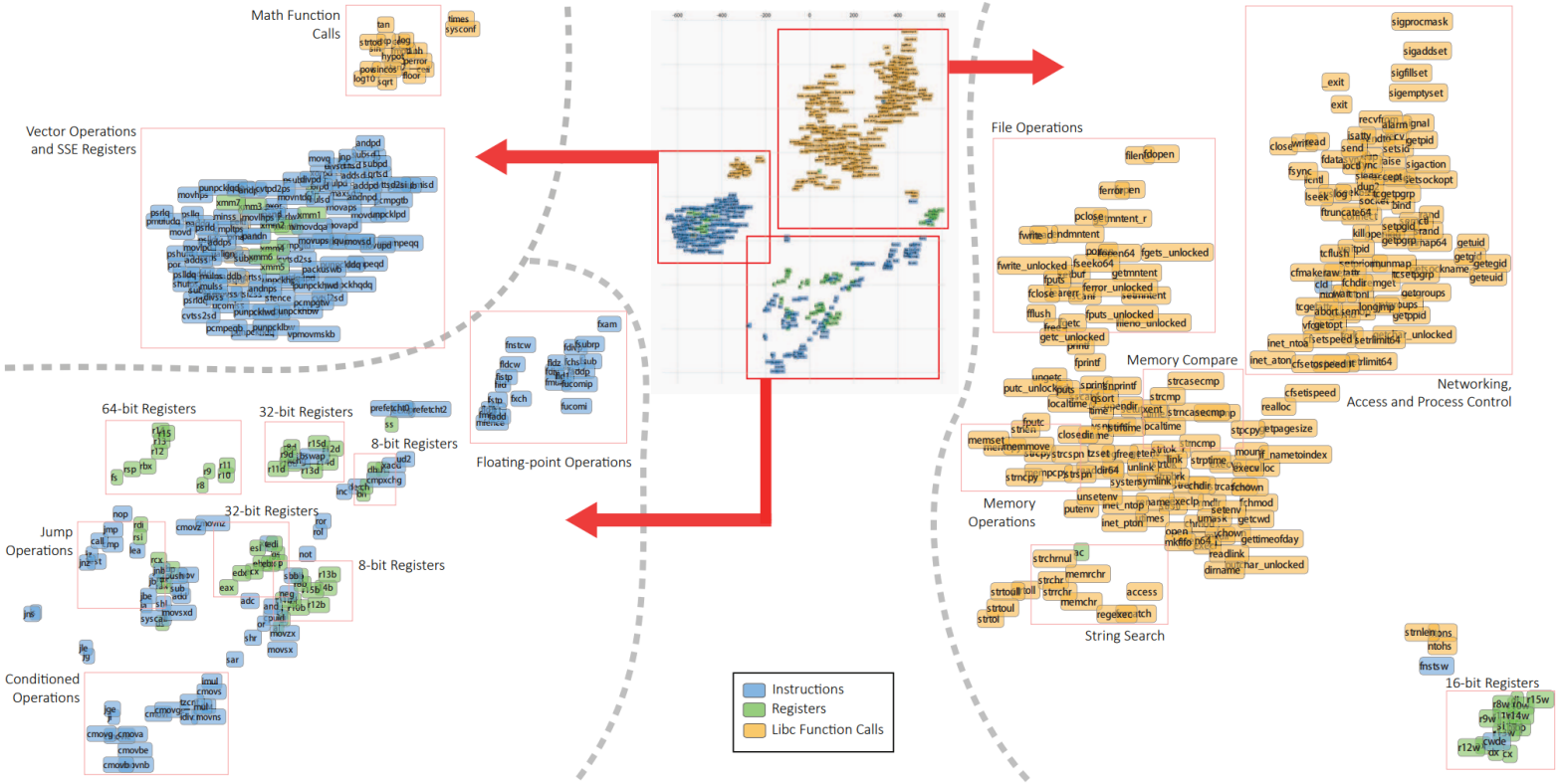
实验结果

- 同源漏洞搜索（高级混淆）：
  - 使用高级混淆器Tigress对漏洞函数进行混淆
    - Encode Literal / Virtualization / JIT
  - 当同时应用三种高级混淆方式时，模型搜索失败

Searching with Obfuscation Options in Tigress									
Name	Heartbleed	ShellshocK	Venom	Clobberin' Time	Shellshock #2	ws-snmp	wget	ffmpeg	avg.
CVE	2014-0160	2014-6271	2015-3456	2014-9295	2014-7169	2014-4877	2014-4877	2015-6826	
# of Positives (k)	15	9	6	10	3	7	3	7	
Encode Literal	100%	77.8%	100%	100%	100%	100%	100%	100%	97.2%
Virtualization	0%	0%	100%	20%	100%	0%	66.7%	0%	35.8%
JIT Execution	53.3%	0%	83.3%	30%	33.3%	0%	0%	100%	37.5%

优劣分析

- 优势
  - 模型对控制流图改变鲁棒
- 劣势
  - 无法跨架构
  - 可解释性差

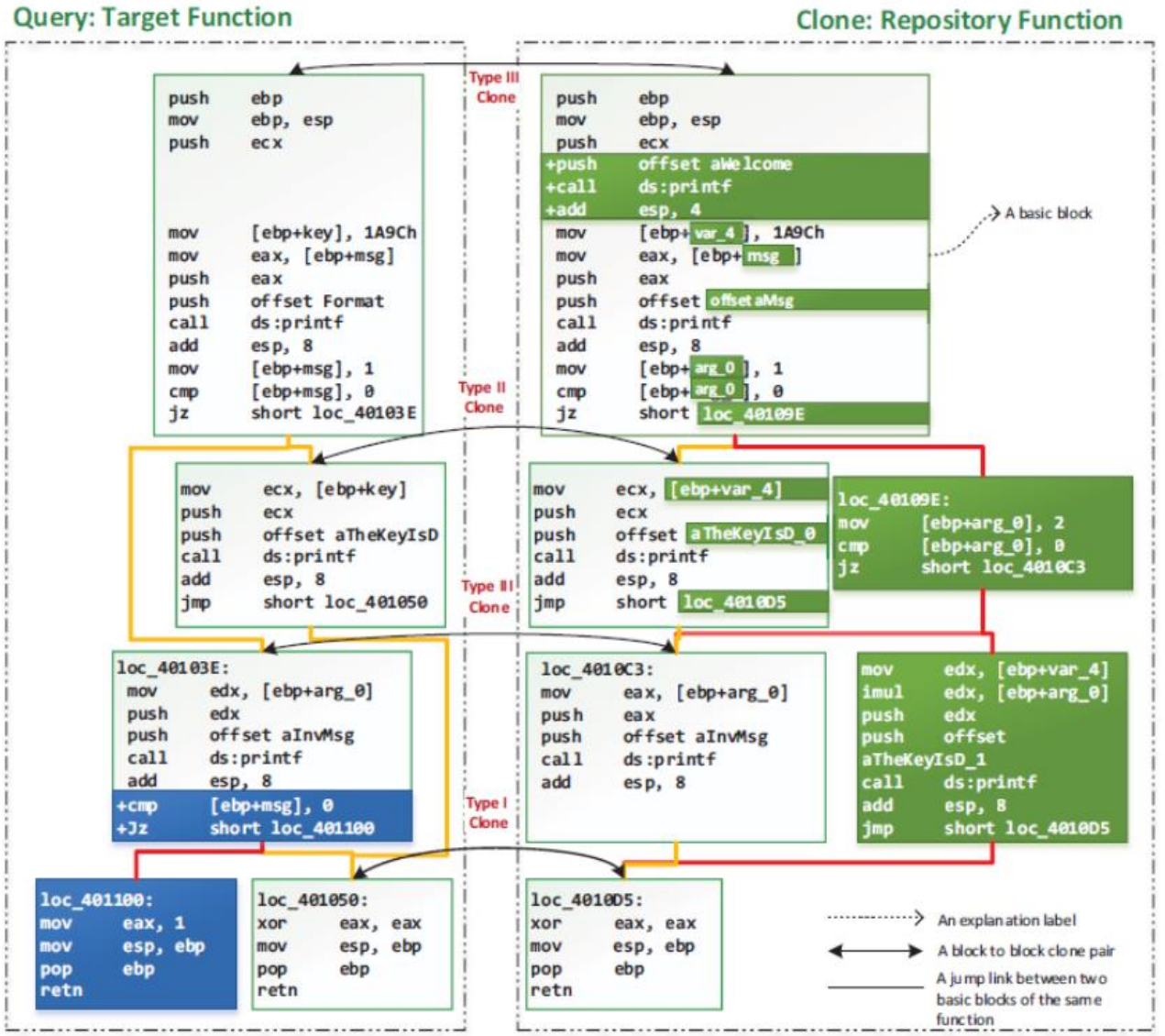


汇编指令向量表示可视化

应用总结

应用总结

- 同源漏洞挖掘
- 软件开源成分分析
- 代码抄袭检测



搜索结果展示

参考文献

[1] Yu L, Lu Y, Shen Y, et al. BEDetector: A Two-Channel Encoding Method to Detect Vulnerabilities Based on Binary Similarity[J]. IEEE Access, 2021, 9: 51631-51645.

[2] Ding S H H, Fung B C M, Charland P. Asm2vec: Boosting static representation robustness for binary clone search against code obfuscation and compiler optimization[C]//2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019: 472-489.

[3] https://www.anquanke.com/post/id/82891 蜻蜓FM涉嫌诈骗投资人和广告主源代码剖析

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。