



DATACON 2020 僵尸网络方向答辩



答辩队伍



上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

1

难点与挑战

2

关键技术研究

3

系统实现

4

总结与展望





难点与挑战



1. 现有工作的不足:

- 传统方法对于同源跨架构二进制代码分析**精度低**:

- 例如 CG, CFG, ACFG 等控制流相关特性

- 依赖深度网络的方法对样本**要求高**:

- 需要大量有标签的样本
- 需要样本具有代表性与普遍性

少量样本下的较高
精度的特征工程?

难点与挑战



2. 数据集的特征:

- Rodata 段被**抹去**，符号表被 **strip**
 - 特征字符串和符号信息等直接信息缺失
- 编译器之间的差异
 - Function prologue和function epilogue存在差异
 - 内联函数的处理不同
- 一些源码中仅存在字符串长度的差异
 - 编译得到的binary控制流相似度高

有限信息和噪声下的有效特征选择？



研究背景与意义



关键技术研究



系统实现



总结与展望

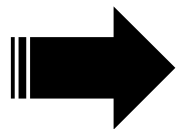


关键技术——用户定义函数识别



静态链接中

大量libc函数



噪声遮盖用户源代码特征

 借助编译后用户定义函数地址连续性，筛出用户定义函数



 借助函数签名进行libc函数识别，剔除libc函数



关键技术

青

大



借助编

其他



借助函

; Attributes: info_from_lumina

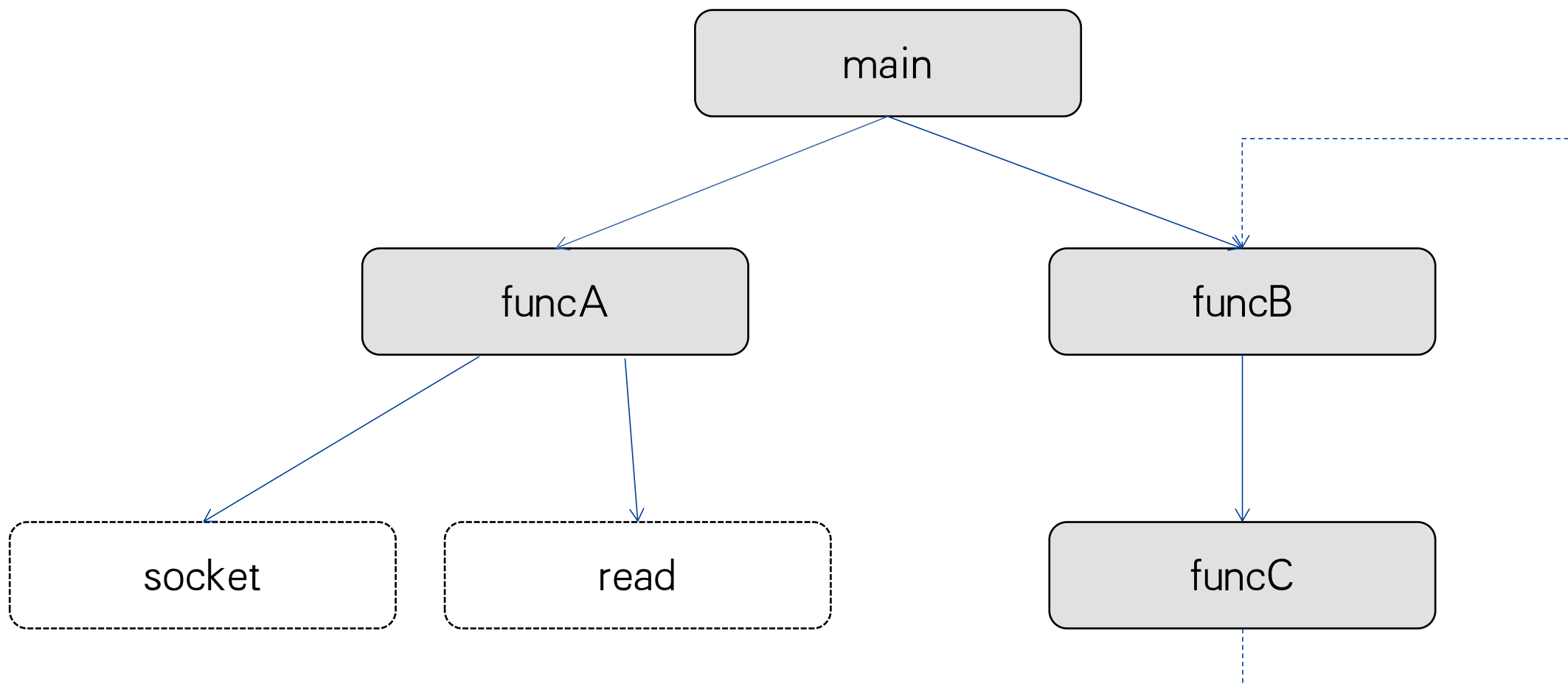
; int __cdecl fcntl(int fd, int cmd, struct flock *lock, char)
fcntl proc nearvar_C= dword ptr -0Ch
fd= dword ptr 4
cmd= dword ptr 8
lock= dword ptr 0Ch
arg_C= byte ptr 10hpush edi ; Alternative name is '__GI__libc_fcntl'
push ebx
sub esp, 14h
mov ecx, [esp+1Ch+cmd] ; cmd
lea eax, [esp+1Ch+arg_C]
mov [esp+1Ch+var_C], eax
mov edi, [esp+1Ch+fd]
mov edx, [esp+1Ch+lock] ; lock
lea eax, [ecx-0Ch]
cmp eax, 2
ja short loc_804F603loc_804F603:
push ebx
mov ebx, edi ; fd
mov eax, 37h ; '7'
int 80h ; LINUX - sys_fcntl
pop ebx
cmp eax, 0FFFFFF00h
mov ebx, eax
jbe short loc_804F623

fcntl:

var_10= -0x10
var_8= -8
var_s0= 0
var_s4= 4
arg_8= 0x10
arg_C= 0x14li \$gp, (off_4442F0+0x7FF0 - .)
addu \$gp, \$t9
addiu \$sp, -0x28
sw \$ra, 0x20+var_s4(\$sp)
sw \$s0, 0x20+var_s0(\$sp)
sw \$gp, 0x20+var_10(\$sp)
addiu \$v1, \$a1, -0x21
addiu \$v0, \$sp, 0x20+arg_C
sltiu \$v1, 3
la \$t9, sub_402430
sw \$v0, 0x20+var_8(\$sp)
sw \$a3, 0x20+arg_C(\$sp)
beqz \$v1, loc_4023F0
sw \$a2, 0x20+arg_8(\$sp)loc_4023F0:
li \$v0, 0xFD7
syscall
la \$t9, sub_4006B0
beqz \$a3, loc_402418
move \$s0, \$v0



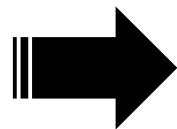
关键技术——用户定义函数Call Graph的建立



关键技术——rodata段的恢复



部分binary中仅字
符串长度存在差异



代码段无法体现完整特征



遍历sections，找到被抹掉的rodata段的范围

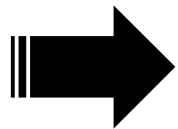
LOAD	00400000	00400094
.init	00400094	00400120
.text	00400120	00403EB0
.fini	00403EB0	00403F0C
.ctors	00444000	00444008
.dtors	00444008	00444010
LOAD	00444010	00444020
.data	00444020	004442F0
.got	004442F0	004444E0
.sbss	004444E0	004444E8
LOAD	004444E8	004444F0
.bss	004444F0	00446638

LOAD	00400000	00400094
.init	00400094	00400120
.text	00400120	00403EB0
.fini	00403EB0	00403F0C
roooodata	00403F0C	00444000
.ctors	00444000	00444008
.dtors	00444008	00444010
LOAD	00444010	00444020
.data	00444020	004442F0
.got	004442F0	004444E0
.sbss	004444E0	004444E8
LOAD	004444E8	004444F0
.bss	004444F0	00446638

关键技术——rodata段的恢复



部分binary中仅字符串长度存在差异



代码段无法体现完整特征



借助IDA的分析结果，获取rodata段中常量的数量、长度和使用位置

```
roooodata:00403F0C # =====
roooodata:00403F0C
roooodata:00403F0C # Segment type: Pure data
roooodata:00403F0C .data # roooodata
roooodata:00403F0C .align 4
roooodata:00403F10 byte_403F10: .space 8 # DATA XREF: sub_4002D0+60to
roooodata:00403F18 byte_403F18: .space 0x30 # DATA XREF: sub_4002D0+178to
roooodata:00403F48 dword_403F48: .space 4 # DATA XREF: sub_4002D0+A8to
roooodata:00403F48 # sub_4002D0+ACto
roooodata:00403F4C dword_403F4C: .space 4 # DATA XREF: sub_4002D0+B4to
roooodata:00403F50 byte_403F50: .space 1 # DATA XREF: sub_4002D0+BCto
roooodata:00403F51 .align 2
roooodata:00403F54 dword_403F54: .space 4 # DATA XREF: sub_4002D0+FCto
roooodata:00403F54 # sub_4002D0+100to
roooodata:00403F58 dword_403F58: .space 4 # DATA XREF: sub_4002D0+108to
roooodata:00403F5C unk_403F5C: .space 1 # DATA XREF: sub_4002D0+120to
roooodata:00403F5D .space 1
roooodata:00403F5E
roooodata:00403F5F
roooodata:00403F60
roooodata:00403F61
roooodata:00403F62
roooodata:00403F63
roooodata:00403F64
roooodata:00403F65
roooodata:00403F66
roooodata:00403F67
roooodata:00403F68 .space 1
roooodata:00403F69
roooodata:00403F6A
roooodata:00403F6B
roooodata:00403F6C
roooodata:00403F6D
roooodata:00403F6E
roooodata:00403F6F
roooodata:00403F70
roooodata:00403F71
roooodata:00403F72
roooodata:00403F73
roooodata:00403F74
roooodata:00403F75
roooodata:00403F76
roooodata:00403F77
roooodata:00403F78
roooodata:00403F79
roooodata:00403F7A
roooodata:00403F7B
roooodata:00403F7C
roooodata:00403F7D
roooodata:00403F7E
roooodata:00403F7F
roooodata:00403F80
roooodata:00403F81
roooodata:00403F82
roooodata:00403F83
roooodata:00403F84
roooodata:00403F85
roooodata:00403F86
roooodata:00403F87
roooodata:00403F88
roooodata:00403F89
roooodata:00403F8A
roooodata:00403F8B
roooodata:00403F8C
roooodata:00403F8D
roooodata:00403F8E
roooodata:00403F8F
roooodata:00403F90
roooodata:00403F91
roooodata:00403F92
roooodata:00403F93
roooodata:00403F94
roooodata:00403F95
roooodata:00403F96
roooodata:00403F97
roooodata:00403F98
roooodata:00403F99
roooodata:00403F9A
roooodata:00403F9B
roooodata:00403F9C
roooodata:00403F9D
roooodata:00403F9E
roooodata:00403F9F
roooodata:00403FA0
roooodata:00403FA1
roooodata:00403FA2
roooodata:00403FA3
roooodata:00403FA4
roooodata:00403FA5
roooodata:00403FA6
roooodata:00403FA7
roooodata:00403FA8
roooodata:00403FA9
roooodata:00403FAA
roooodata:00403FAB
roooodata:00403FAC
roooodata:00403FAD
roooodata:00403FAE
roooodata:00403FAF
roooodata:00403FB0
roooodata:00403FB1
roooodata:00403FB2
roooodata:00403FB3
roooodata:00403FB4
roooodata:00403FB5
roooodata:00403FB6
roooodata:00403FB7
roooodata:00403FB8
roooodata:00403FB9
roooodata:00403FBA
roooodata:00403FBB
roooodata:00403FBC
roooodata:00403FBD
roooodata:00403FBE
roooodata:00403FBF
roooodata:00403FC0
roooodata:00403FC1
roooodata:00403FC2
roooodata:00403FC3
roooodata:00403FC4
roooodata:00403FC5
roooodata:00403FC6
roooodata:00403FC7
roooodata:00403FC8
roooodata:00403FC9
roooodata:00403FCA
roooodata:00403FCB
roooodata:00403FCC
roooodata:00403FCD
roooodata:00403FCE
roooodata:00403FCF
roooodata:00403FD0
roooodata:00403FD1
roooodata:00403FD2
roooodata:00403FD3
roooodata:00403FD4
roooodata:00403FD5
roooodata:00403FD6
roooodata:00403FD7
roooodata:00403FD8
roooodata:00403FD9
roooodata:00403FDA
roooodata:00403FDB
roooodata:00403FDC
roooodata:00403FDD
roooodata:00403FDE
roooodata:00403FDF
roooodata:00403FE0
roooodata:00403FE1
roooodata:00403FE2
roooodata:00403FE3
roooodata:00403FE4
roooodata:00403FE5
roooodata:00403FE6
roooodata:00403FE7
roooodata:00403FE8
roooodata:00403FE9
roooodata:00403FEA
roooodata:00403FEB
roooodata:00403FEC
roooodata:00403FED
roooodata:00403FEE
roooodata:00403FEF
roooodata:00403FF0
roooodata:00403FF1
roooodata:00403FF2
roooodata:00403FF3
roooodata:00403FF4
roooodata:00403FF5
roooodata:00403FF6
roooodata:00403FF7
roooodata:00403FF8
roooodata:00403FF9
roooodata:00403FFA
roooodata:00403FFB
roooodata:00403FFC
roooodata:00403FFD
roooodata:00403FFE
roooodata:00403FFF
roooodata:00404000
```

Direct	Op	Address	Text
Up	r	sub_4002D0+A8	lw \$a0, (dword_403F48 - 0x400000)((\$v1))
Up	o	sub_4002D0+AC	addiu \$v1, (dword_403F48 - 0x400000)

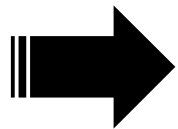
Line 1 of 2

OK Cancel Search Help

关键技术——binary跨架构特征构建



由于不同架构的编译器处
理不同（函数内联等）



CFG存在差异

 我们选取用户定义全局变量的访问作为核心特征

```
0x4002d0
['unk_403F10', 'unk_403F18', 'dword_403F48', 'dword_403F48', 'dword_403F4C', 'byte_403F50', 'dword_403F54', 'dword_403F54', 'dword_403F58', 'unk_403F5C']
0x400d70
['dword_403FC0', 'dword_403FC0']
0x401720
['unk_403FE0', 'unk_403FE8', 'unk_403FEC']
0x401e34
['unk_403FF0']
0x400120
['unk_403FFC', 'unk_440000', 'unk_440000']
0x4001dc
['unk_403FFC', 'unk_440000', 'unk_440000']
0x4007e0
['unk_440000']
0x401ebc
['unk_440000']
0x403630
['unk_440000', 'unk_440000', 'unk_440000', 'unk_440000']
0x403e40
['unk_440000']
Function 0x0 cannot be identified!
Counter()
{4210448: 8, 4210456: 48, 4210504: 4, 4210508: 4, 4210512: 4, 4210516: 4, 4210520: 4, 4210524: 100, 4210624: 32, 4210656: 8, 4210664: 4, 4210668: 4, 4210672: 12, 4210684: 245764, 4456448: 16384}
```



研究背景与意义



关键技术研究



系统实现

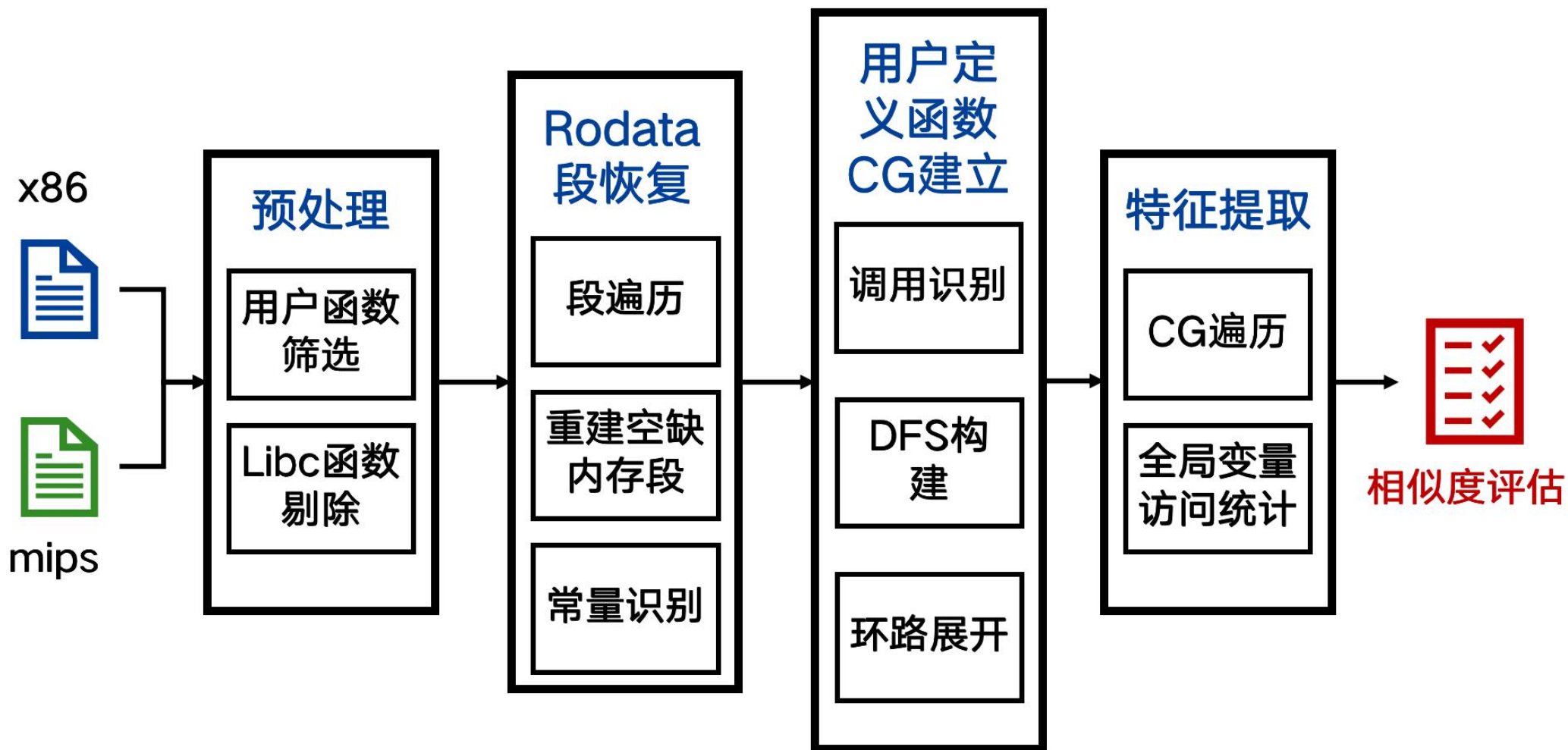


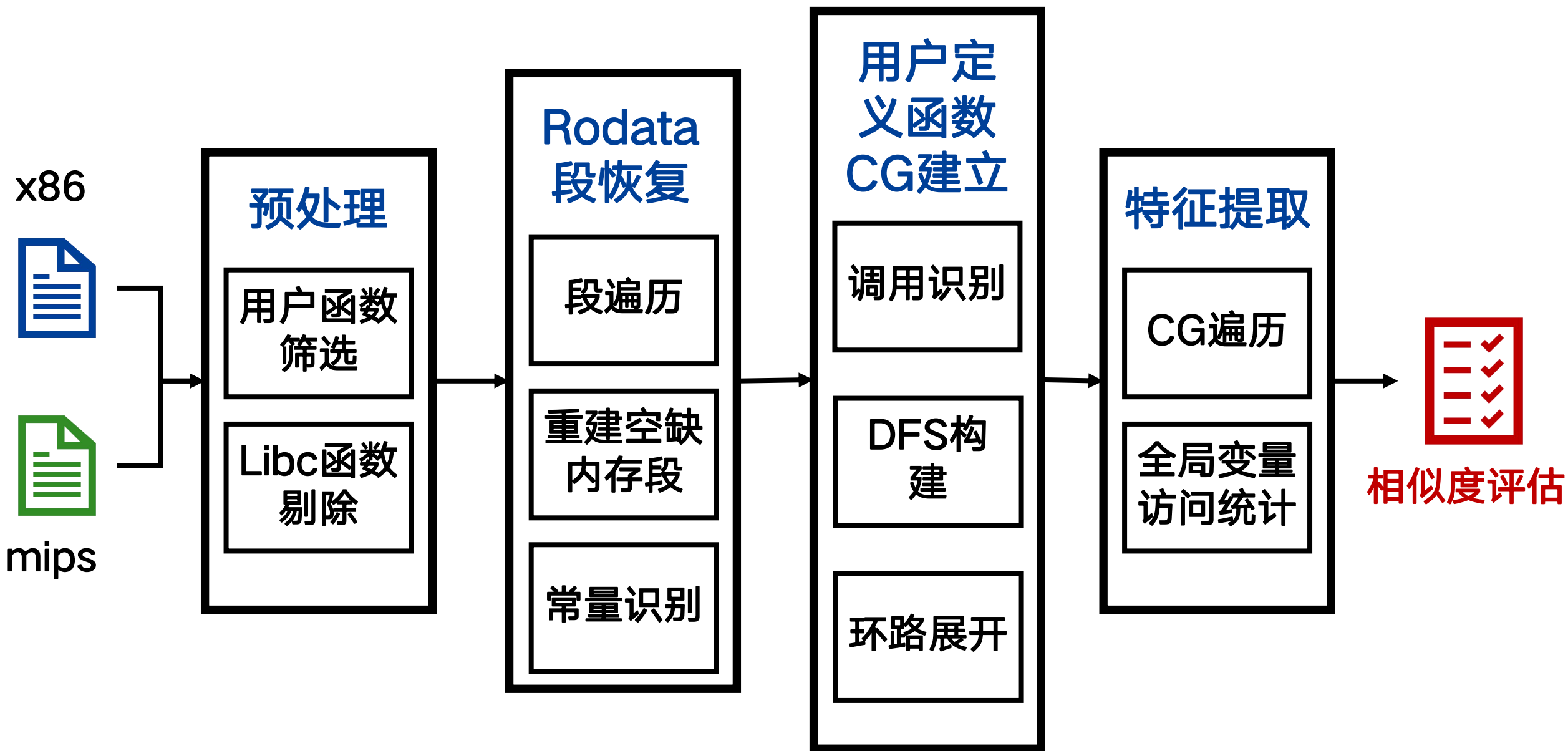
总结与展望





跨框架同源二进制识别流程







研究背景与意义



关键技术研究



系统实现



总结与展望





工作总结与展望



- 本项目中我们使用user define functions识别、建立Call Graph、全局变量统计还原的方法对同源跨架构的binary进行了相似性分析，以达到细粒度的同源配对分析。
- 由于时间原因，我们在研究过程中还有许多ideas没能实践...

Ideas



- 将不同架构的binary提升到IR，再在IR上进行程序分析，以提高对多架构binary的适应性
- 同架构下两个binary代码相似性聚类的简单方法：在fileA中多次选取数条地址无关指令，在fileB中寻找是否出现过该地址无关指令，以此计算两个binary的代码相似度。
- 对单一函数进行动态检查（黑盒），如果能保证该函数及其子调用不会有外部输入（全局变量、input、syscall），那么它的运行结果应该由其输入参数决定，我们可以写一个wrapper，构造好传入参数，以不同的输入多次调用该函数，使该函数成功执行，若两个binary中的该函数返回值相同，则可以认为这两个函数相同

谢谢！

