机器语言大模型赋能软件自主可控与安全可信

朱文宇
清华大学

中国电机工程学会
CHINESE SOCIETY FOR ELECTRICAL ENGINEERING

背景-软件是网络空间的基石

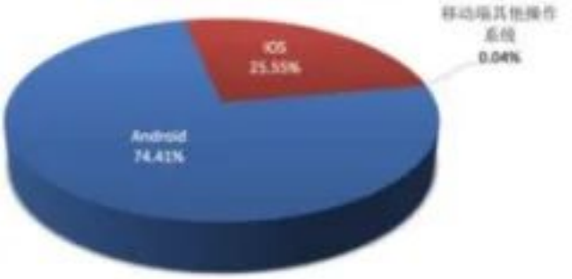
自主可控

安全可信

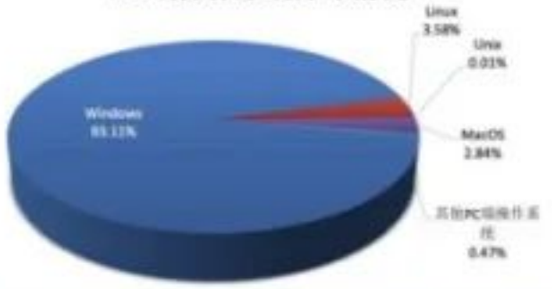
背景-需求1：软件生态面临自主可控难题

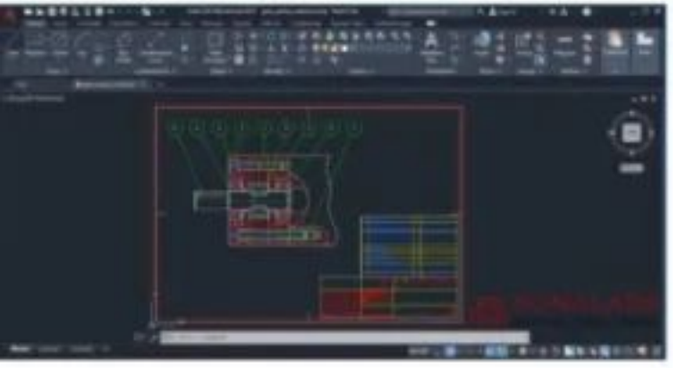
关键软件长期由国外主导

移动端操作系统分布情况



PC端操作系统分布情况





工业设计软件

CAD（计算机辅助设计）软件

CAE（计算机辅助工程验证）软件

CAM（计算机辅助制造）软件

工业控制软件

GE/西门子、MES、DCS、SCADA

面临断供、安全、知识产权风险

XCodeGhost (2015)

工业信息管理软件

产品：ERP、CRM、SCM

企业：SAP、Oracle、Saleforce

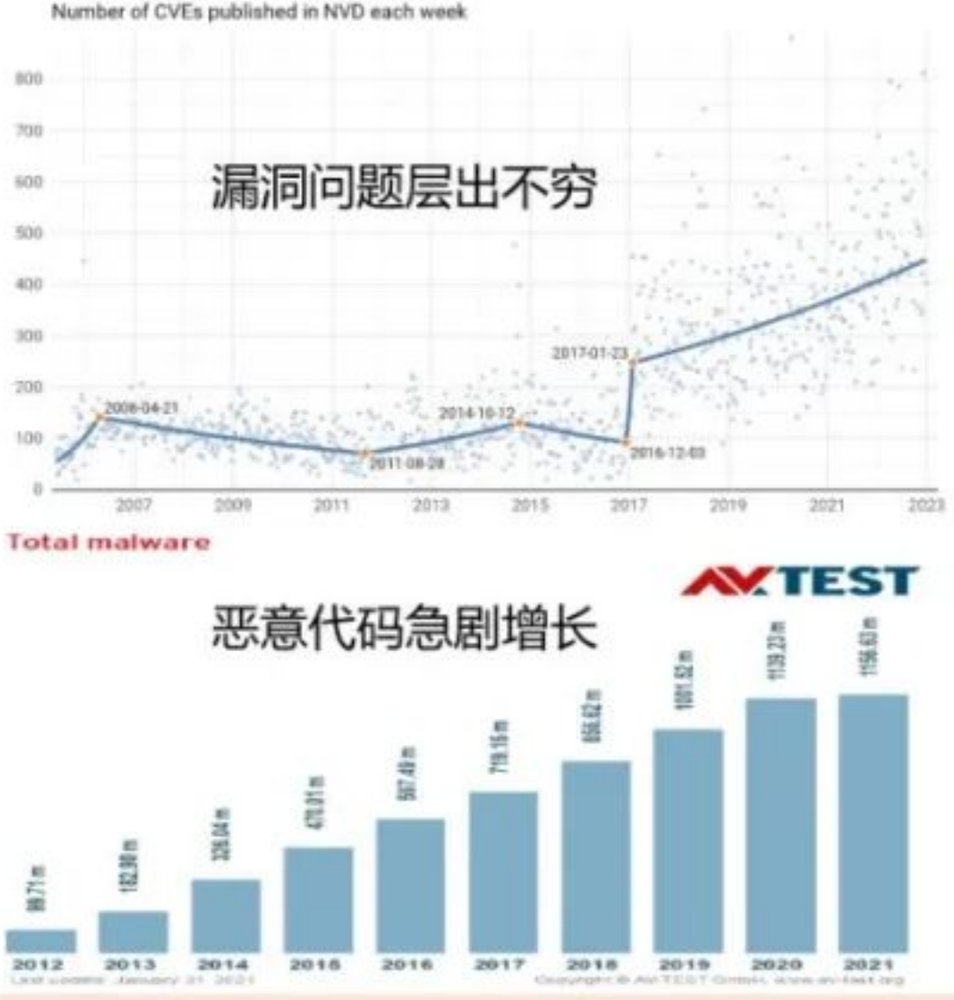
SolarWinds (2020)

xz (2024)

挑战：关键软件闭源，供应链风险高，自主可控难度大

背景-需求2：软件生态面临安全可信难题

软件安全风险未知



成为网络攻击的重要突破口

MIRAI 僵尸网络

震网病毒(伊朗核设施)

iPhone 越狱 & 安卓 Root

电网断电（乌克兰）

WannaCry (勒索150+国家)

挑战：目标软件闭源，分析难度大，安全问题隐藏深

关键问题-软件开发与安全

自主可控国产化替代

自主可控

- 软件分析理解
- 软件设计开发
- 软件测试部署

安全风险发现与防范

安全可信

- 软件分析理解
- 发现安全问题
- 防范安全问题

关键问题是分析、理解目标（闭源）软件

02 关键挑战1：软件分析——二进制信息缺失

源代码 calc.c

```c
#include <stdio.h>
#include <stdlib.h>

void answer(char *name, int x){
  printf("Hey %s, the answer is: %d\n", name, x);
}
void calc(int y){
  return y*y;
}
void main(int argc, char *argv[]){
  int x = 40 + atoi(argv[2]);
  int y = calc(x);
  answer(argv[1], y);
}
```

中间表示IR

```
define i8* @answer(i8*,
      i8*) #0 {
  %3 = call i64
      @strlen(i8* %0) #3
  %4 = call i64
      @strlen(i8* %1) #3
  %5 = add i64 %3, 1
  %6 = add i64 %5, %4
  %7 = call noalias i8*
      @malloc(i64 %6) #4
  %8 = call i8*
      @strcpy(i8* %7,
      i8* %0) #4
  %9 = getelementptr
      inbounds i8, i8*
      %7, i64 %3
  %10 = call i8*
      @strcpy(i8* %9,
      i8* %1) #4
  ret i8* %7
}
```

汇编码

```
answer:
.LFB5:
  .cfi_startproc
  pushq %rbp
  .cfi_def_cfa_offset 16
  .cfi_offset 6, -16
  movq  %rsp, %rbp
  .cfi_def_cfa_register 6
  subq  $16, %rsp
  movq  %rdi, -8(%rbp)
  movl  %esi, -12(%rbp)
  movl  -12(%rbp), %edx
  movq  -8(%rbp), %rax
  movq  %rax, %rsi
  leaq  .LC0(%rip), %rdi
  movl  $0, %eax
  call  printf@PLT
  nop
  leave
  .cfi_def_cfa 7, 8
  ret
  .cfi_endproc
```

机器码

```
chao @ calc  $ xxd calc
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0300 3e00 0100 0000 8005 0000 0000 0000  ..>.............
00000020: 4000 0000 0000 0000 7819 0000 0000 0000  @.......x.......
00000030: 0000 0000 4000 3800 0900 4000 1d00 1c00  ....@.8...@.....
00000040: 0600 0000 0400 0000 4000 0000 0000 0000  ........@.......
00000050: 4000 0000 0000 0000 4000 0000 0000 0000  @.......@.......
00000060: f801 0000 0000 0000 f801 0000 0000 0000  ................
00000070: 0800 0000 0000 0000 0300 0000 0400 0000  ................
00000080: 3802 0000 0000 0000 3802 0000 0000 0000  8.......8.......
```

符号、类型、边界等信息逐步优化/丢弃

02 关键挑战1：软件分析——二进制信息缺失

源代码 calc.c

```c
#include <stdio.h>
#include <stdlib.h>

void answer(char *name, int x){
  printf("Hey %s, the answer is: %d\n", name, x);
}
void calc(int y){
  return y*y;
}
void main(int argc, char *argv[]){
  int x = 40 + atoi(argv[2]);
  int y = calc(x);
  answer(argv[1], y);
}
```

反编译

机器码

```
chao @ calc  $ xxd calc
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0300 3e00 0100 0000 8005 0000 0000 0000  ..>.............
00000020: 4000 0000 0000 0000 7819 0000 0000 0000  @.......x.......
00000030: 0000 0000 4000 3800 0900 4000 1d00 1c00  ....@.8...@.....
00000040: 0600 0000 0400 0000 4000 0000 0000 0000  ........@.......
00000050: 4000 0000 0000 0000 4000 0000 0000 0000  @.......@.......
00000060: f801 0000 0000 0000 f801 0000 0000 0000  ................
00000070: 0800 0000 0000 0000 0300 0000 0400 0000  ................
00000080: 3802 0000 0000 0000 3802 0000 0000 0000  8.......8.......
```

lifting

中间表示IR

```
define i8* @answer(i8*,
      i8*) #0 {
  %3 = call i64
      @strlen(i8* %0) #3
  %4 = call i64
      @strlen(i8* %1) #3
  %5 = add i64 %3, 1
  %6 = add i64 %5, %4
  %7 = call noalias i8*
      @malloc(i64 %6) #4
  %8 = call i8*
      @strcpy(i8* %7,
      i8* %0) #4
  %9 = getelementptr
      inbounds i8, i8*
      %7, i64 %3
  %10 = call i8*
      @strcpy(i8* %9,
      i8* %1) #4
  ret i8* %7
}
```

反汇编

汇编码

```
answer:
.LFB5:
  .cfi_startproc
  pushq %rbp
  .cfi_def_cfa_offset 16
  .cfi_offset 6, -16
  movq  %rsp, %rbp
  .cfi_def_cfa_register 6
  subq  $16, %rsp
  movq  %rdi, -8(%rbp)
  movl  %esi, -12(%rbp)
  movl  -12(%rbp), %edx
  movq  -8(%rbp), %rax
  movq  %rax, %rsi
  leaq  .LC0(%rip), %rdi
  movl  $0, %eax
  call  printf@PLT
  nop
  leave
  .cfi_def_cfa 7, 8
  ret
  .cfi_endproc
```

无中生有，恢复缺失的信息

关键挑战2：软件理解——人在回路

源代码 calc.c

```c
#include <stdio.h>
#include <stdlib.h>

void answer(char *name, int x){
  printf("Hey %s, the answer is: %d\n", name, x);
}
void calc(int y){
  return y*y;
}
void main(int argc, char *argv[]){
  int x = 40 + atoi(argv[2]);
  int y = calc(x);
  answer(argv[1], y);
}
```

① 结构分析

② 语义分析

代码功能分析：
内存分配、
加密解密？
敏感数据如何流动？

严重依赖人工经验

③ 安全分析 功能分析 性能分析

机器码

```
chao @ calc  $ xxd calc
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0300 3e00 0100 0000 8005 0000 0000 0000  ..>.............
00000020: 4000 0000 0000 0000 7819 0000 0000 0000  @.......x.......
00000030: 0000 0000 4000 3800 0900 4000 1d00 1c00  ....@.8...@.....
00000040: 0600 0000 0400 0000 4000 0000 0000 0000  ........@.......
00000050: 4000 0000 0000 0000 4000 0000 0000 0000  @.......@.......
00000060: f801 0000 0000 0000 f801 0000 0000 0000  ................
00000070: 0800 0000 0000 0000 0300 0000 0400 0000  ................
00000080: 3802 0000 0000 0000 3802 0000 0000 0000  8.......8.......
```

漏洞　恶意代码　逆向破解　供应链

软件转写　性能优化

信息缺失

人在回路

软件分析

软件理解

安全分析

功能开发

智能化解决方案：大语言模型

信息缺失

软件分析

信息自动补偿

人在回路

软件理解

人类意图对齐

安全分析

功能开发

内容自动生成

现有大语言模型难以分析二进制程序（闭源软件）

- 软件可以抽象为三个维度的表示

自然语言

人类
需求
意图

设计
实现

源代码

```
#include <stdio.h>
#include <stdlib.h>
void answer(char *name, int
x){
  printf("Hey %s, the answer
is: %d\n", name, x);
}

void main(int argc, char
*argv[]){
  int x = 40 + atoi(argv[2]);
  answer(argv[1], x);
}
```

编译
部署

二进制程序（机器语言）

```
00 00 00 00 00 00
AF 21 F4 31 CD 1F
21 00 50 11 F0 03
70 23 1B 7B B2 20
CD 1B 00 FE 53 CA
BE AF 21 2C 31 77
D0 11 98 03 CD 8F
31 CD 1F 31 32 F7
CB 52 22 27 31 3E
32 DE 31 CD 52 41
32 D9 31 AF 32 DB
32 DD 48 21 00 00
```

加载
运行

程序执行（机器语言）

进程
(时刻1)

进程
(时刻N)

ChatGPT

通用大模型

code llama

代码大模型

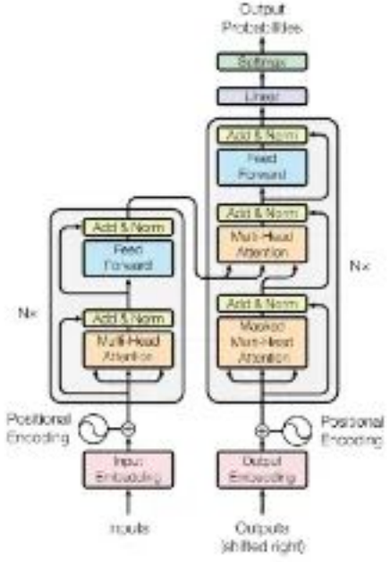
MLM
Machine Language Model
0110
1011
0101

机器语言是网络空间基石，但是缺少智能化解决方案

方案简介：大语言模型

Transformer架构

（自注意力机制）



海量算力

（1000+ 英伟达卡*天）

亿元算力

适量算力

（10+ 英伟达卡*天）

百万算力

少量算力

（8张4090卡）

十万算力

预训练

（厂商）

微调

（专业用户）

推理

（普通用户）

提示词工程

RAG 知识增强

工具使用

海量数据

（TB级高质量数据）

适量标注数据

（MB级高质量数据）

目标数据

（按需）

模型优化

（面向二进制软件）

专业数据自动生成

（源代码、二进制、文本、二进制标注数据）

数据

- 大规模机器语言-自然语言-源代码多模态对齐数据
- 100TB规模，业界公开数据<100GB

技术

- 全自动数据生成、标注、对齐
- 优化模型设计，深刻理解机器语言，对齐人类专家

工程

- 自研机器语言模型训练方法
- 完善的机器语言模型基础设施

Conference
ISSTA
ISSTA: International Symposium on Software Testing and Analysis

IEEE S&P

USENIX SECURITY SYMPOSIUM

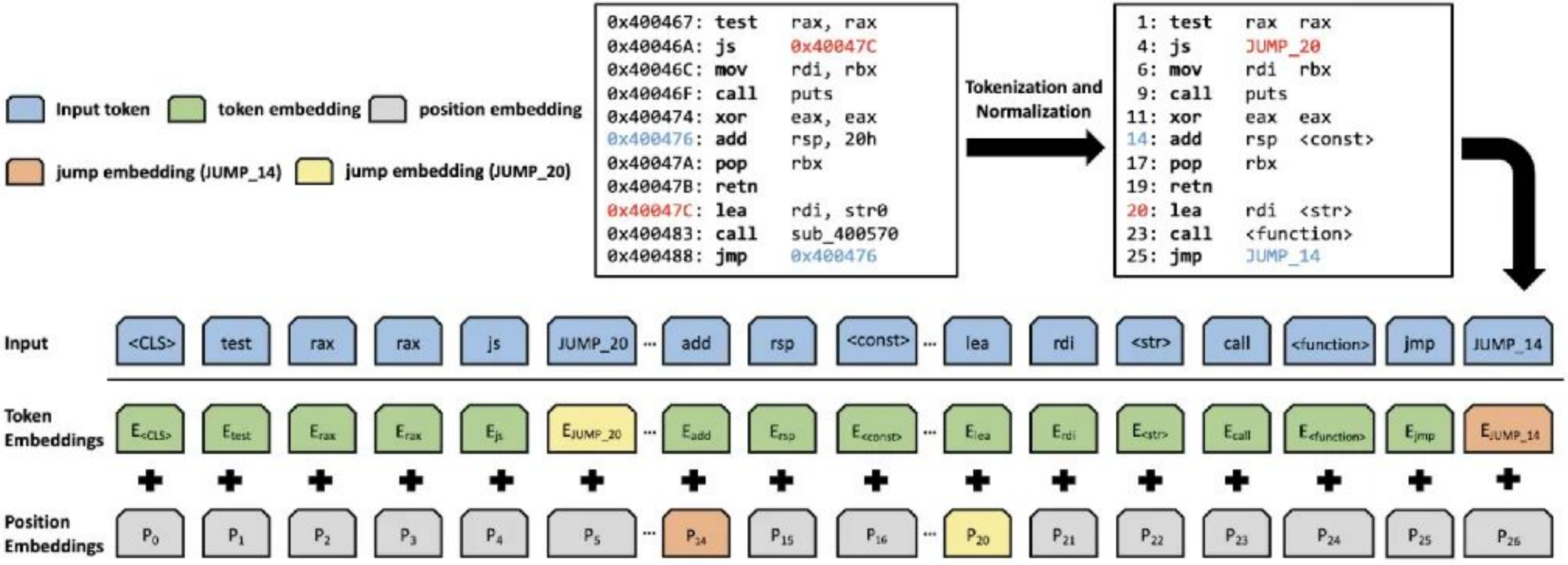
NDSS

acm SIGSAC

多个关键技术零的突破

关键技术1：融合领域知识的模型优化

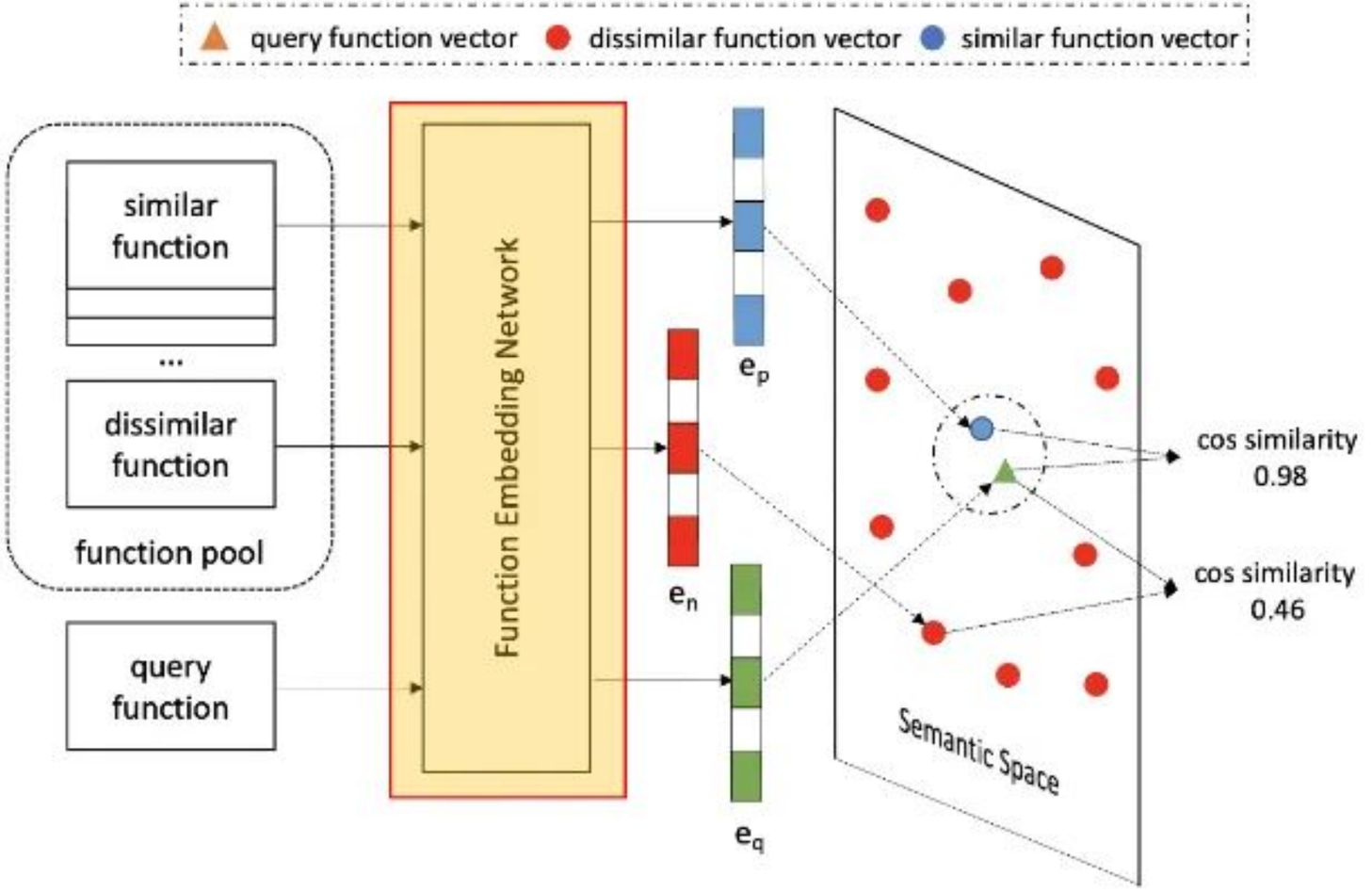
修改模型设计，融入代码领域知识（指令语义、跳转关系等）



"jTrans: Jump-Aware Transformer for Binary Code Similarity Detection." ISSTA 2022

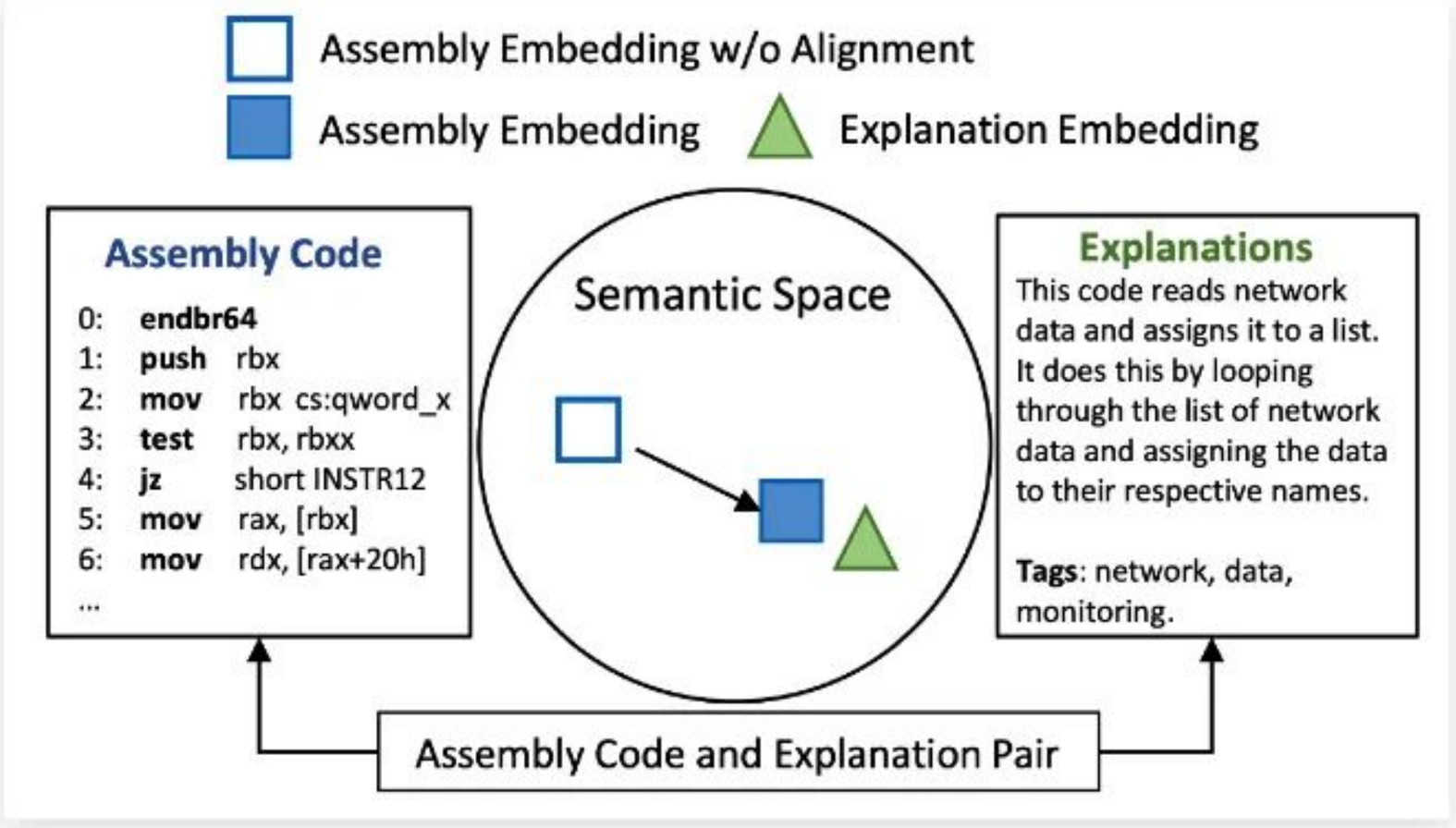
关键技术2：基于对比学习的语义理解

利用对比学习技术，使得语义相似的二进制代码embedding接近



关键技术3：基于多模态学习的语义理解

利用多模态技术，将语义空间与人类意图对齐，更准确地表示二进制代码语义



"CLAP: Learning Transferable Binary Code Representations with Natural Language Supervision." ISSTA 2024

我们的解决方案：机器语言大模型MLM

多平台

MLM
Machine Language Model

0 1 1 0
1 0 1 1
0 1 0 1

多架构

多场景
多能力

安全分析、性能优化、功能拓展

供应链分析 | 漏洞攻防 | 版权保护 | APT分析 | 工具生成 | 性能优化 | 功能翻译 | 软件迁移

全场景智能化，开创软件分析新范式

语义分析

语义摘要 → 语义搜索 | 功能分类 | 语义比较 | 函数命名 | 函数类型 | 输入格式

语义理解超越人类专家水平，高效辅助逆向分析

结构分析

指令边界 | 函数边界 | 跳转关系 → 反汇编 | 控制流图 | 函数调用图 | 反编译

反汇编核心功能全面超越IDA Pro

1 背景

2 关键问题

3 智能化方案

4 典型应用

5 总结

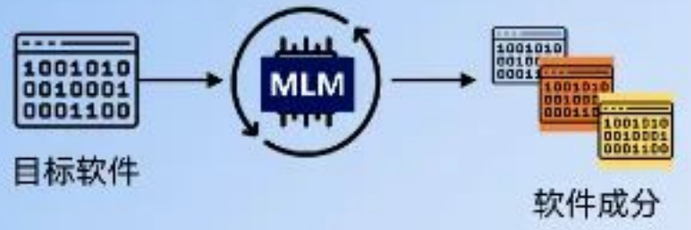
软件逆向分析

突破卡脖子技术

软件生态迁移

信创国产化、老旧软件升级迁移

软件供应链分析



细粒度、高速、语义对齐的二进制代码比对

软件一致性检测

解决采购痛点

漏洞挖掘

大模型赋能0day、1day漏洞挖掘

软件版权保护分析

破解取证难题

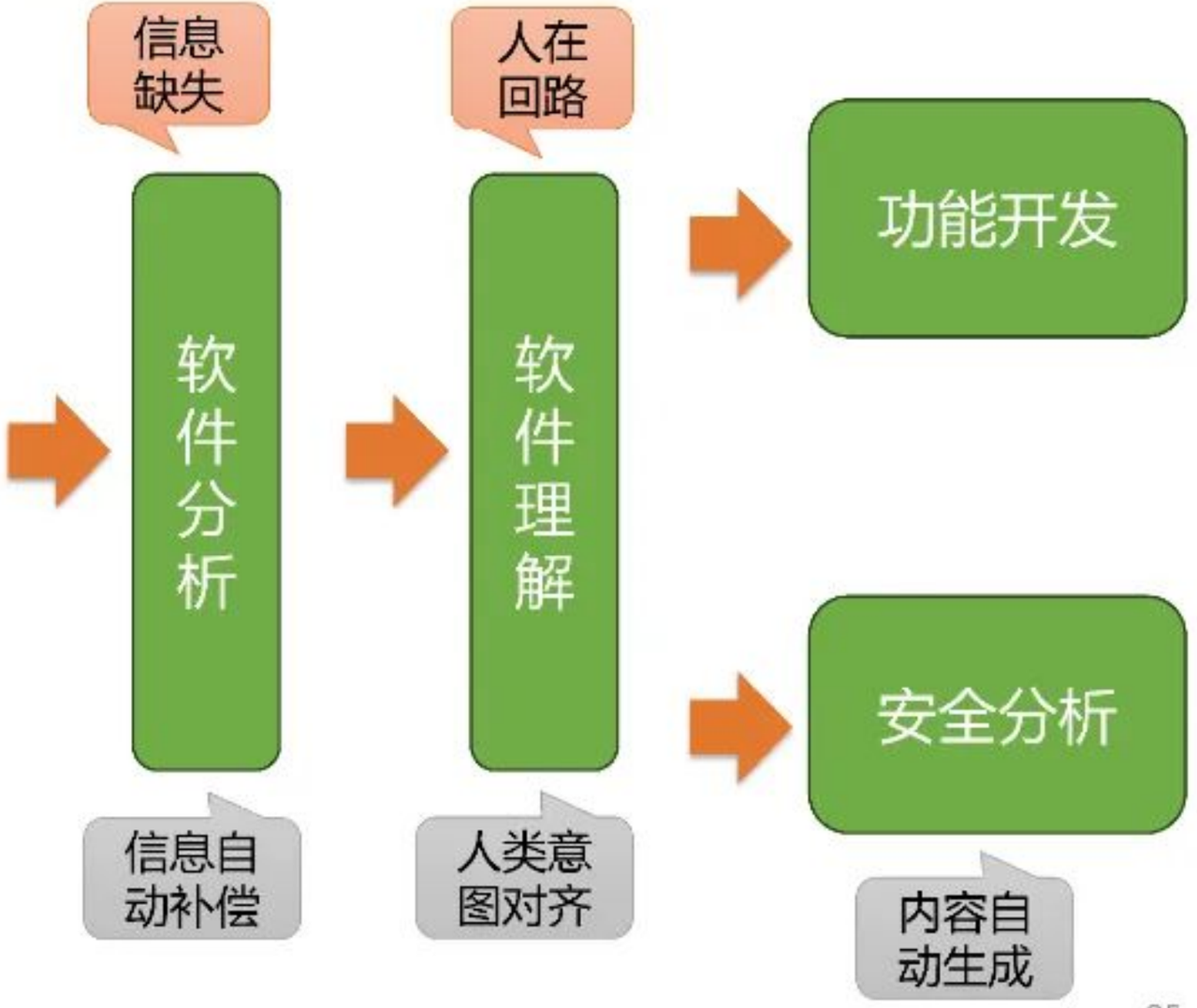
Demo-颠覆工作模式的智能逆向分析

mlm01.com

- 将黑盒二进制程序变成白盒代码
- 首次实现像人类专家一样理解二进制程序语义
- 将专家从繁琐的底层代码分析中解放出来，专注高层分析任务

软件自主可控、安全可信

解决方案：大语言模型



机器语言大模型MLM（全球首个）

多平台

MLM
Machine Language Model

0110
1011
0101

多架构

多场景
多能力

安全分析、性能优化、功能拓展

供应链分析 | 漏洞攻防 | 版权保护 | APT分析 | 工具生成 | 性能优化 | 功能翻译 | 软件迁移

全场景智能化，开创软件分析新范式

语义分析

语义摘要 → 语义搜索 | 功能分类 | 语义比较 | 函数命名 | 函数类型 | 输入格式

语义理解超越人类专家水平，高效辅助逆向分析

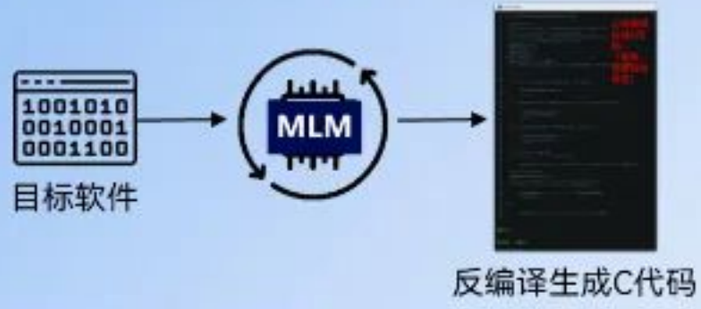
结构分析

指令边界 | 函数边界 | 跳转关系 → 反汇编 | 控制流图 | 函数调用图 | 反编译

反汇编核心功能全面超越IDA Pro

广泛的应用场景

软件逆向分析

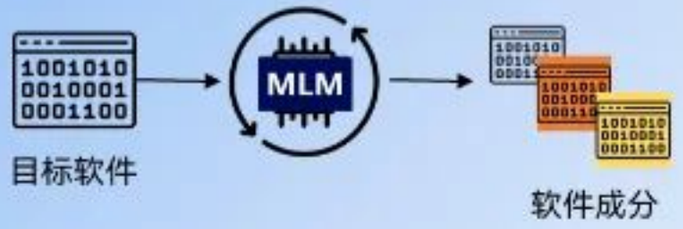


突破卡脖子技术

软件生态迁移

信创国产化、老旧软件升级迁移

软件供应链分析



细粒度、高速、语义对齐的二进制代码比对

软件一致性检测

解决采购痛点

漏洞挖掘

大模型赋能0day、1day漏洞挖掘

软件版权保护分析

破解取证难题

拥抱人工智能
智胜网络空间

https://mlm01.com