NASAC 2019原型工具竞赛

规则检测竞赛的题目及要求

1. 简介

本竞赛拟针对C/C++语言编写的软件系统中的程序缺陷问题开展自动化检测工具的评比。本次竞赛将综合基准测试集实测和PPT报告进行评分。

此次竞赛由华为技术有限公司进行奖励赞助。优秀竞赛成果未来可能被应用于华为公司的代码缺陷检测系统，提升软件系统的质量。

1. 题目及解释

本竞赛拟针对15类C/C++程序中的缺陷规则进行检测，缺陷规则编号和描述可参见下表。

|  |  |
| --- | --- |
| 规则编号 | 规则描述 |
| 1 | switch里面的变量类型和case中的类型不匹配，比如enum和int混用  enum DAY {  MON=1, TES, THU  };  int main() {  enum DAY day;  switch(day) {  case 1:  return 1;  default:  return 0;  }  } |
| 2 | 空指针解引用  int \*p = NULL;  int q = \*p; |
| 3 | 除零错，除（/）或模（%）运算的第二个操作数可能为0  int g(int x) {  x = x - 5;  int y = 100/x;  if(y > 0) {  int z = 100 / y;  }  } |
| 4 | 数组下标越界  int a[10];  a[10] = 0; |
| 5 | custodial指针变量可能没被释放或返回，一个custodial指针变量是指它是唯一一个指向它的存储区域的指针  char \*p = new char[20]; // p是一个custodial指针  char \*q = p; // p不再是custodial指针  p = new char[20]; // p又变为custodial指针  q = p + 0; // p还是custodial指针，指针的算术运算不改变custodial属性  strcpy(p, “hello”) // p不再是custodial指针，将指针作为函数参数，会改变custodial属性 |
| 6 | 通过下标访问字符数组'\0'之后的内容  char buf[20];  strcpy( buf, "a" );  char c = buf[4]; // 访问’\0’之后的未赋值内容 |
| 7 | 指针变量double free  int \*p = malloc(sizeof(int));  free(p);  free(p); |
| 8 | 声明有返回类型的函数在某些路径上没有return返回值  int f(int x) {  if(x > 5) {  return x;  }  }  // if条件不成立时没有返回值 |
| 9 | 比较运算中混合使用有符号数与无符号数  int a = -1;  unsigned int b = 1;  if (a > b)  由于a > b表达式会先将a强制转换为unsigned int，所以a > b为真，不符合用户预期 |
| 10 | 成员变量域长度太小  enum color { red, green, yellow, blue, black};  struct abc { enum color c:2; };  enum color定义了5个枚举量，而abc中的变量c的大小只有2bit，最多只能表示4个变量 |
| 11 | switch表达式是枚举类型时，至少一个枚举常量不存在case中，同时，没有提供default分支  enum DAY {  MON=1, TES, THU  };  int main() {  enum DAY day;  switch(day) {  case 1:  return 1;  }  return 0;  } |
| 12 | 类成员函数返回值是指向成员的非const指针或引用，但成员的访问级比这个函数要低  class X {  private:  int a;  public:  int \*f() { return &a; }  };  a是private变量，而public函数f返回了a的地址，外部可以通过函数f来修改a |
| 13 | 类的const成员函数返回非const的指针  class X {  int \*p;  int \*f() const { return p; }  };  函数f声明为const函数，但返回成员指针变量p，就能够通过p修改p指向内容的值 |
| 14 | 类的非静态指针成员变量未在析构函数中释放  class X {  int \*p;  ~X() {  }  }  指针变量p没有在析构函数中释放，可能存在内存泄漏 |
| 15 | 将指向继承类数组的指针赋值给指向基类的指针  Derived \*d = new Derived[10];  Base \*b;  b = d; // Warning  b = &d[0]; //OK  Base是基类，Derived是继承类，b = d之后如果访问b[i]，实际上是访问  i \* sizeof(Base)，而不是i \* sizeof(Derived)，可能出错 |

1. 解题要求
2. 采用静态分析方法进行检测；竞赛之前我们将给出基准测试集，包括源代码工程和编译脚本；
3. 竞赛会于11月20日开放竞赛最终数据集供参赛队测试，参加复赛的队伍需准备一份关于参赛工具的技术报告（PPT形式）；

（3）不依赖商业工具，且需列出原型工具所使用的开源协议及所依赖的开源项目；

（4）工具报告的输出格式

以JSON格式作为输出格式，满足如下格式：

{

"TotalDefects": 1, // 总共检测到的缺陷数量

"Defects":[

{

"DefectType": 1, // 整型，代表缺陷编号

"FileName": "/path/to/buggy/file.c", // 字符串类型，代表缺陷所在文件路径

"Line": 100 // 整型，代表缺陷所在行号

}

]

}

}

1. 评分标准
2. 基准测试集实测占80%，PPT报告占20%；
3. 其中基准测试集实测主要考虑检出率（40%）、误报率（40%）、分析性能（20%）三个方面。