NASAC 2020原型工具竞赛

代码缺陷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。

1. 题目及解释

本竞赛拟针对12类C/C++程序中的代码缺陷进行检测，缺陷编号和描述可参见下表。

|  |  |
| --- | --- |
| 缺陷编号 | 缺陷描述 |
| 1 | 指针的偏移越界：unsigned long PorcMsg（void \*pMsg, unsigned long ulTypeId, unsigned long \*pulLen）{ MSG\_STRU \*pstMsg = （MSG\_STRU \*）pMsg; unsigned long ulParaNum = ntol（pstMsg->ulParaNum）; unsigned long ulLoop; ... unsigned long ulLen; unsigned char \*pucData; ... if（ulParaNum > MSG\_MAX\_NUM） { return -1; } pucData =（unsigned char \*）pstMsg + sizeof（MSG\_STRU） - 1; ... for（ulLoop = 0; ulLoop < ulParaNum; ulLoop++） { unsigned long ulTmpType = \*（unsigned long \*）pucData; pucData += sizeof（unsigned long）; ...... pucData += sizeof（unsigned long）; if（ulTmpType != ulType） { pucData += ulLen;//外部报文中的长度进行报文地址的偏移时，由于没有校验读取位置仍然位于报文缓冲区之内 continue; } ... } return -1;} |
| 2 | 外部数据直接应用于循环条件，造成数组下标访问越界。for（i =0;i < pstOcts->nLen; i++）  {  pstBigInt->aVal[i] = pstOcts->octs[i];  } |
| 3 | 转换结构体前未校验，直接访问域，造成越界while (udwOffset < udwLen) { psIe = (S\_MSG\*)((unsigned char \*)pPureData + udwOffset); if(FALG\_RECOVERY == psIe->ucType) { \*pucRecoveryValue=(unsigned char)(\*((unsigned char\*)psIe+ sizeof(S\_MSG))); .... }} |
| 4 | 外部数据直接运算后整数溢出，从而导致指针访问越界 while (ucType != 0) { ucLen = (\*(unsigned char\*)(pucMsg + (\*pulLen))); if (0 == ucLen) { ... return -1; } (\*pulLen) += (ucLen \* 4 - 1);ucType = \*((unsigned char\*)(pucMsg + (\*pulLen))); ... } |
| 5 | 死循环PSR\_UINT32 ulTotalLen=0; /\* 初值为0\*/while（ulTotalLen < ulLeftPktLen） {/\* 溢出为0后，解析从头开始\*/ switch（ucType）{ ... case TYPE\_1: ... ulTotalLen += ulLeftDataSize; /\*ulLeftChunkDataSize外部数据运算前未校验，可能溢出为0\*/ if（ulTotalLen >= ulLeftPktLen）{ ... bExitLoop = TRUE; break; } ... pucCurPos += ulLeftDataSize; ulLeftLen -= ulLeftDataSize; …  } // switch  if（TRUE == bExitLoop） { break; }} // while |
| 6 | 除零错，除（/）或模（%）运算的第二个操作数可能为0int g(int x) { x = x - 5; int y = 100/x;if(y > 0) { int z = 100 / y;}} |
| 7 | 不可信数据拼接导致SQL注入char buffer[100]={0};char \*string = "select password from author order by "; char \*string1 ="'";//SQL语句拼接不可信参数if(-1 ！= springtf\_s(buffer,100,"%s,%s,%s,%s",string,string1,name,string1)){retcode = SQLAllocHandle(SQL\_HANDLE\_STMT, hdbc, &hstmt);retcode = SQLPrepare(hstmt,&buffer, SQL\_NTS);//准备sql查询,name为不可信输入retcode = SQLExecute(hstmt);//执行SQL命令}else{/\*handle error\*/} |
| 8 | 运算过程中可能导致溢出或反转unsigned char g\_num\_add =0;unsigned char g\_num\_sub =255;void incNum(){ g\_num\_add++;//可能溢出或者反转. }void decNum(){ g\_num\_sub--;//可能溢出或者反转. }int main(){ incNum(); decNum();  return;} |
| 9 | 空指针解引用int \*p = NULL;int q = \*p; |
| 10 | 访问已经释放的内存struct node{ int value; struct node \*next;};void free\_list(struct node \*head){ for (struct node \*ptr = head; ptr != NULL; ptr = ptr ->next) /\* 【错误】 解引用已经释放的内存 \*/ { free(ptr); }} |
| 11 | 引用容器前要确保容器元素存在bool NoCompliant(const NodeKeyList &srcList, const NodeKeyList &snkList){ NodeKey srcNode = srcList.front(); NodeKey snkNode = snkList.back(); /\* ...do something... \*/} |
| 12 | 对文件路径进行规范化后再使用 char \*fileName = GetMsgFromRemote(); ... sprintf\_s(untrustPath, sizeof(untrustPath), "/tmp/%s", fileName); char \*text = ReadFileContent(untrustPath);  |

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. 竞赛会于复赛前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。