# 程序说明

* vul
  + 命名：stack\_overflow
  + 漏洞：缓冲区溢出漏洞
  + 攻击：代码注入攻击
  + Shellcode：写文件
  + 本次代码注入攻击是借助跳板指令实现的，详细原理见代码注入课件
  + 攻击模式为单纯的输入数据攻击，为了保证兼容性，跳板指令的地址是在读入文

件时自动构造

* + #define SIZE 200 定义缓冲区大小
  + #define MAX\_LENGTH 216 定义偏移大小，即从缓冲区起始位置到返回地址的偏 移
  + 以下代码是用来构造跳板指令的地址

if(length==MAX\_LENGTH){

long long int tmp=(long long int)jmp\_rsp+4;

for(int i=0;i<8;++i){

v=(tmp%16)+(tmp/16%16)\*16;

buf[length++] = v;

tmp=tmp/16/16;

}

}

* Input
  + code\_injection.txt此文件为shellcode，其功能为向output文件夹中的code\_injection\_output.txt文件中写入字符串buffer\_overflow\_gyj\_001 attack success
  + hello\_world\_normal.txt此文件为正常输入
* output
  + 如果攻击成功会产生code\_injection\_output.txt文件，文件内容为buffer\_overflow\_gyj\_001 attack success
  + 如果攻击失败，将不产生文件，或文件为空
* check
  + 命名：stack\_overflow\_check
  + 获取output文件夹下code\_injection\_output.txt文件
  + 如果此文件不存在，或为空，返回失败信息，否则将此文件中的信息返回
* compile.sh define.jsonssss
  + 编译命令和配置信息已经写入compile.sh和define.json文件
* shellcode 生成
  + 用汇编写一段满足攻击要求的程序hello.asm
  + 按照如下命令生成shellcode

nasm -felf64 hello.asm -o hello64.o

ld -s -o hello64 hello64.o

for i in $(objdump -d hello64 |grep "^ " |cut -f2); do echo -n '\x'$i; done; echo