# 代码注入原理：

代码注入技术利用系统的漏洞，将攻击者写好的代码注入到进程空间中，通过劫持控制流，攻击者可以直接执行这部分代码以实现恶意功能。代码注入攻

击一般借助于缓冲区溢出等方法，将恶意代码写入攻击的系统中。缓冲区溢出是指计算机从外部接受输入后，向进程缓冲区内填充数据时超过缓冲区本身的容量，溢出的数据会覆盖原有系统中的地址或数据，改变程序执行的控制流。

通过往程序的[缓冲区](http://baike.baidu.com/view/266782.htm" \t "http://baike.baidu.com/_blank)写超出其长度的内容，造成缓冲区的溢出，从而破坏程序的[堆栈](http://baike.baidu.com/view/93201.htm" \t "http://baike.baidu.com/_blank)，造成程序崩溃或使程序转而执行其它指令，以达到攻击的目的。造成[缓冲区](http://baike.baidu.com/view/266782.htm" \t "http://baike.baidu.com/_blank)溢出的原因是程序中没有仔细检查用户输入的参数。

# 分析过程：

通过gdb附加调试漏洞程序找到eip在栈中的位置以及偏移。

1.编译漏洞程序：gcc -fno-stack-protector -z execstack -g -o ggtest ggtest.c其中

-fno-stack-protector和-z execstack这两个参数会分别关掉DEP

和Stack Protector

sudo sysctl -w kernel.randomize\_va\_space=0关闭ASLR

2.运行漏洞程序，等待连接：./ggtest

3.启动gdb附加调试：ps -ax | grep ggtest查看进程号pid

gdb ggtest pid 启动gdb对漏洞程序进行附加调试

4.运行攻击程序: ./run.py 发送正常的数据(AAAA)

5.继续gdb调试： next命令直到运行到函数testVul()

step命令进入函数testVul()

X/100xw $rsp打印栈中的信息

info f查看栈的基本信息，从中可以看出被压入栈中rip的值，

以及rip在栈中的位置，还有buff在栈中的地址

通过攻击程序发送的数据可以知道str，buff在栈中的位置，

从而计算出rip到str的偏移offset

1. 构造shellcode：shellcode的地址为buff地址+offset+8，用shellcode的地址

覆盖rip，从而构造shellcode=’A’\*offset+shellcode地址

+shellcode代码

运行：

1.漏洞程序ggtest通过vulnerabilities/stack\_overflow/run.py调用执行，执

行命令./run.py;

2.执行攻击程序exploits/stack\_overflow/run.py，发送所构造的shellcode，

达到栈溢出攻击（读出/etc/passwd的内容）

# 注意事项：

1. 在对栈溢出漏洞进行攻击时，由于连接没有释放，导致执行完一次后不

能马上再执行（会出现绑定失败错误），可以等一会再执行或者重新打开

终端或者修改端口号。

2. 如果对漏洞程序进行了修改，只需改攻击程序的ret参数即可。