wget https://github.com/mudongliang/source-packages/raw/master/CVE-2018-9138/binutils-2.29.tar.gz

tar -xvf binutils-2.29.tar.gz

cd binutils-2.29

CFLAGS="-g -fsanitize=address" LDFLAGS="-fsanitize=address -ldl" ./configure

make

安装

然后下载poc，运行

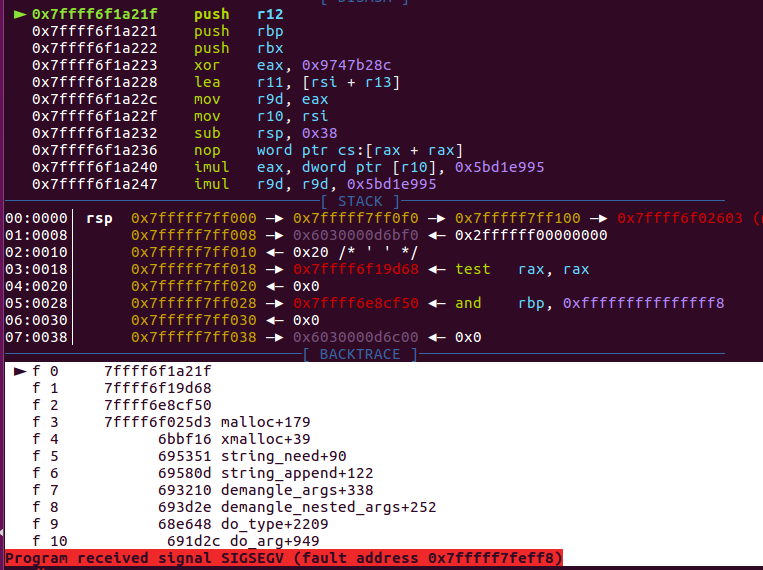
./binutils/cxxfilt < poc

程序崩溃，显示

SUMMARY: AddressSanitizer: stack-overflow ??:0 ??

==35308==ABORTING

Gdb挂起程序

程序段错误

Vmmap观察栈区可以发现，栈已经到栈顶了。

0x7fffff7ff000 0x7ffffffff000 rw-p 800000 0 [stack]

猜测这是个不断压栈，导致程序崩溃的漏洞。



那么bt查看一下调用栈，找到循环的开始

可以看到demangle\_template这个函数调用的do-type出现问题

这个工具主要是用来将函数名 demangle。

# [mangle和demangle](http://www.cnblogs.com/robinex/p/7892795.html)

         C/C++语言在编译以后，函数的名字会被编译器修改，改成编译器内部的名字，这个名字会在链接的时候用到。

         将C++源程序标识符(original C++ source identifier)转换成C++ ABI标识符(C++ ABI identifier)的过程称为mangle；相反的过程称为demangle。

而cxxfilt 这个工具是将编译器修改的函数恢复用的。

通过向上追踪输入流可以看到，

/\* otherwise, value parameter \*/

/\* temp is initialized in do\_type \*/

success = do\_type (work, mangled, &temp);

string\_delete(&temp);

if (!success)

break;

if (!is\_type)

{

s = &param;

程序最终是在这里调用了触发漏洞的函数

我们设置条件断点b do\_type if \*\*mangled=='F'来进行研究

在do\_type中，如果检测到内容字符串为f，那么这是一个不认识的东西，就进入demangle\_nested\_args

然后进入demangle\_args 来试图确认这个未识别的东西是什么，最终进入do\_args函数，do\_args中再次调用do\_type确定这东西是什么。

漏洞分析：

这里函数并没有限制深度，也就是说如果我们设置一个超长的f字符串，会导致不断压栈，因此报错