# 生成库文件

## 文件

* 一个头文件：libfunc.h

头文件中声明使用的函数，如加法myADD（），乘法表myMUL（）两个函数，还有一个test函数，在test函数中根据k的每一项的值是0或1来决定如何调用加法和乘法这两个函数。

* 一个c文件：libfunc.c

C文件中定义myADD（），myMUL（）两个函数，这两个函数不能定义在一起（挨着写），可能由于空间预取，如果连续定义这两个函数，在实验结果中会有干扰，调用加法时会看到乘法所在的内存也被加载到cache中。所以在定义时，myADD（）和myMUL（）函数之间要插入写一个无关的函数，将他们隔开。

此外这两个函数中加入延迟（for循环），使得他们的执行时间变长，从而能方便观测到。

test函数中定义一个数组k，每一项的值是0或者1。如果为0，就调用myADD函数。否则调用myMUL函数。在每次调用完一个函数后，执行一个延迟（for循环），这样在做出的点状图中可以清楚的区分k的每一个值，而不会粘在一起。

## 编译过程：

1. **gcc -c -g -fPIC -o libfunc.o libfunc.c**

-c：表示只编译不连接

-g：表示保留调试信息，否则在Mastik执行时会提示找不到文件的信息。（到Symbol\_bfd.c中的dwarf\_next\_cu\_header函数的返回值为-1，错误为没有剩余的可遍历的单元。因为读不到调试信息。）

-fPIC：PIC指的是Position Independent Code（位置无关代码）。共享库要求有此选项，以便实现动态连接。编译动态库时，要求产生与位置无关的代码。

1. gcc -shared -g -o libfunc.so libfunc.o

生成共享库：

-shared表示生成一个共享库

-g：表示保留调试信息

# 编译main文件，指定其使用生成的库文件

主文件中include库文件的头文件（libfunc.h）

While（1）循环中调用test函数，一直执行。

## 编译过程

gcc main.c -I /home/becca/Documents/sidechannel/FRsimple/ /home/becca/Documents/sidechannel/FRsimple/libfunc.so -o main -ldl -pthread

-I（此处为大写的i）后面跟的是libfunc.so的头文件所在的目录，以及libfunc.so所在的位置，输出为main。

Ldd main可以查看其连接的库，确认有没有连接到libfunc.so。

如果提示找不到库文件，则需要将库文件所在的文件夹输出到环境变量：

export LD\_LIBRARY\_PATH=（），括号中为库文件所在的绝对路径，如：

export LD\_LIBRARY\_PATH=/home/becca/Documents/sidechannel/FRsimple/

然后再执行程序。

Taskset -c 0 ./main(这里需要设置cpu亲和度，将受害者和攻击者绑在一个处理器的不同核上。具体看逻辑核对应的处理器是哪一个。这个命令表示main在0核上执行)

# Mastik的使用

[https://www.cnblogs.com/sctb/p/13516381.html](https://www.cnblogs.com/sctb/p/13516381.html这里介绍了mastik) 这里有mastik的相关介绍

首先解压Mastik-0.02-AyeAyeCapn：tar -xzvf Mastik-0.02-AyeAyeCapn.tgz。

## 安装依赖

(mastik提供了三种符号转换为文件偏移量的函数。

sym\_loadersymboloffset() 在加载程序符号表中找到一个符号。 sym\_debuglineoffset() 找到对应于特定源代码行的机器代码。在Linux中，这些功能依赖于 libbfd，libdwarf 和 libelf。 要使用，请确保 libdwarf-devel 和 binutils-devel( 或者 libdwarf-dev，binutils-dev 和 libelf-dev) 已安装。）

1. sudo apt-cache search dwarf
2. sudo apt-get install libdwarf-dev
3. sudo apt-cache search bfd
4. sudo apt-get install binutils-dev
5. sudo apt-get install binutils-multiarch-dev
6. 还要安装libelf-dev：

Sudo apt-get install libelf-dev

1. 再安装libwarf：准备已有的文件夹：libdwarf-code-cb7198abde46c2ae29957ad460da6886eaa606ba

进入这个文件夹，然后运行：

1. Sh scripts/FIX-CONFIGURE-TIMES(在运行configure之前的配置，为了fix important timestaps）
2. ./configure (如果permission denied），则可将configure文件的属性修改为可执行）
3. Make

此时Mastik就可以使用了。

## 修改源文件

在执行FR的时候，需要确定阈值（Threshold），slot，还有监视的位置。然后到源文件/Mastik/demo/FR-gnupg-1.4.13.c中去修改。

char \*monitor[] = {

//"mpih-mul.c:85",

//"mpih-mul.c:271",

//"mpih-div.c:356"

libfunc.c:5

libfunc.c:27

// "crypto/ec/ec\_mult.c:452"}；

进入文件夹Mastik-0.02-AyeAyeCapn中，运行：

1. ./configure
2. Make
3. 进入demo(cd demo)
4. 运行
5. taskset -c 2

./FR-gnupg-1.4.13 /home/becca/Documents/sidechannel/FRsimple/libfunc.so

（将攻击者设置在2号核上执行，2和0是在同一个处理器的。）

监视的位置需要找到libfunc.c中的myADD函数和myMUL函数中的一行，记下行号。（libfunc.c文件尽可能删掉没用的代码，包括注释的代码）。

阈值的确定可以使用Mastik的demo文件夹下面的FR-threshold，也可以使用另一个代码（Cache-attack-aes/Flush+Reload/Threshold-Flush+Reload/threshold）来确定阈值。

Slot的选取与系统有关，与监视的位置的多少也有关系，监视的位置变多时，需要相应的增大slot的大小，否则会错过很多slot，导致结果中很多0.一般监视两个位置时，slot选择2000，2500左右。

# FR-gnupg-1.4.13.c的执行流程

fr：这是一个描述符，记录监视的地址，其结构如下：



vl的\*\*data中存放监视的地址。size为固定值16，len中存放监视的地址的个数。

执行过程：

1. fr\_probe() 准备fr的空间
2. 为每个监视的位置找到offset。
3. fr\_monitor()向描述符监视的一组指针中添加一个指针（也就是监视的地址）
4. fr\_pribe():（从磁盘）加载一次监视的地址，取到cache中，记录访问的时间，放在result[i]中，然后冲掉.
5. fr\_trace():
   1. prev\_time=rdtscp64();记录一下当前的时间
   2. fr\_probe:探测一次监视的地址（得到的res中的结果是从内存（mem）中取数据的延迟）
   3. while循环：{
6. prev\_time+slot(等一个slot，等待时钟到达prev\_time+slot这个时间）
7. fr\_probe:再测一次（也就是等待受害者执行后，测cache的状态）

：停止条件：while（is\_active):如果有一个res[i]的值低于阈值（Threshold），说明被害者很可能已经开始执行被监视的程序了，就跳出循环，往下开始监视监视行的执行状态，记录重载的时间并输出。否则就一直测cache的状态。

}

* + 1. while（idle\_count <500;count<100000):（如果空转数大于500或者监视的次数大于10万次，就退出，这里空转指的是missed的次数，也就是slot被错过的次数）

idle\_count++;res+=len;count++;

if missed!=0 ——>res的各项设置为0.

else fr\_probe（） 探测一次cache的状态

if(is\_active(有一个res的值小于阈值，即监视的位置至少有一个被访问了。）

idle\_count=0;

prev\_time+=slot(再过一个slot）

missed=slotwait（prev\_time)：如果prev\_time<当前时间，也就是已经错过了一个或多个slot，返回1（missed），否则就sleep，等待时钟到达prev\_time这个时间。

当探测的行数多的时候，slot也要相应的增加，否则，会miss掉slot，最后res的结果都是全0，然后退出程序。