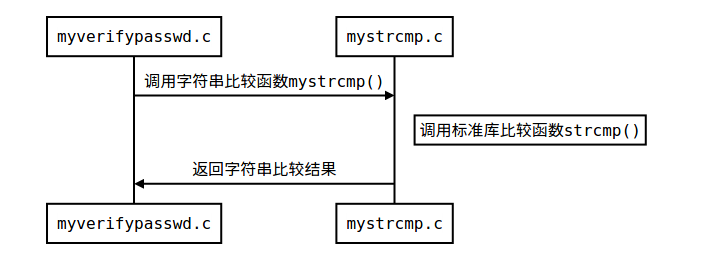
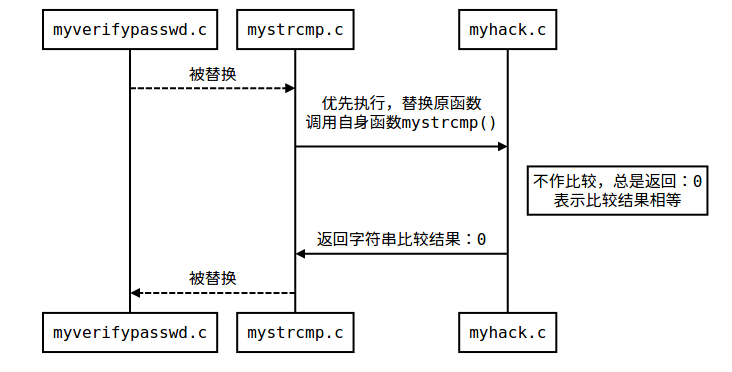
**[LD\_PRELOAD的偷梁换柱之能](http://www.cnblogs.com/net66/p/5609026.html)**

作者： [net66](http://www.cnblogs.com/net66) 原创  
本文网址：<http://www.cnblogs.com/net66/p/5609026.html>  
发布日期：2015 年 06月 22日

**一、LD\_PRELOAD是什么**

LD\_PRELOAD是Linux系统的一个环境变量，它可以影响程序的运行时的链接（Runtime linker），它允许你定义在程序运行前优先加载的动态链接库。这个功能主要就是用来有选择性的载入不同动态链接库中的相同函数。通过这个环境变量，我们可以在主程序和其动态链接库的中间加载别的动态链接库，甚至覆盖正常的函数库。一方面，我们可以以此功能来使用自己的或是更好的函数（无需别人的源码），而另一方面，我们也可以以向别人的程序注入程序，从而达到特定的目的。

**二、程序调用流图**

* LA\_PRELOAD替换前  
  
* LA\_PRELOAD替换后  
  

**三、演示程序代码**

* 主程序

// myverifypasswd.c

#include <stdio.h>

#include <string.h>

#include "mystrcmp.h"

void main(int argc,char \*\*argv) {

char passwd[] = "password";

if (argc < 2) {

printf("usage: %s <password>\n",argv[0]);

return;

}

if (!mystrcmp(passwd,argv[1])) {

printf("Correct Password!\n");

return;

}

printf("Invalid Password!\n");

}

* 调用库

// mystrcmp.h

#include <stdio.h>

int mystrcmp(const char \*s1, const char \*s2);

// mystrcmp.c

#include <stdio.h>

#include <string.h>

#include "mystrcmp.h"

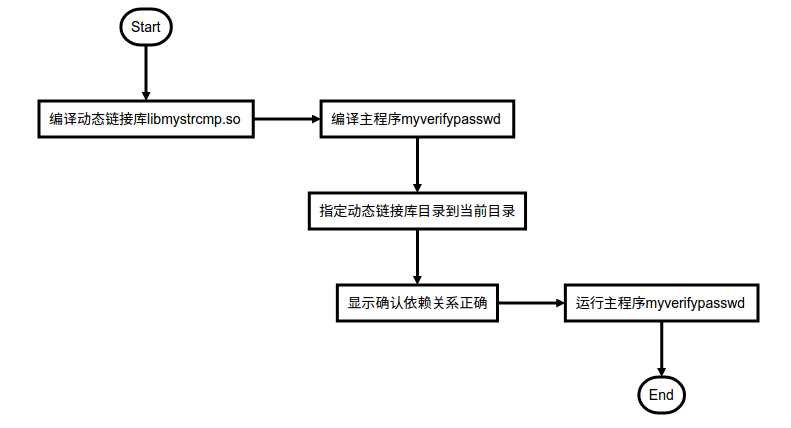
int mystrcmp(const char \*s1, const char \*s2)

{

return strcmp(s1,s2); //正常字串比较

}

**四、程序编译与试验**

* 演示流程图  
  
* 编译、设置指令

gcc mystrcmp.c -fPIC -shared -o libmystrcmp.so #编译动态链接库

gcc myverifypasswd.c -L. -lmystrcmp -o myverifypasswd #编译主程序

export LD\_LIBRARY\_PATH=/home/n6/桌面/LD\_PRELOAD #指定动态链接库所在目录位置

ldd myverifypasswd #显示、确认依赖关系

./myverifypasswd #运行主程序myverifypasswd

* 终端运行结果

n6@X240s:~/桌面/LD\_PRELOAD$ gcc mystrcmp.c -fPIC -shared -o libmystrcmp.so

n6@X240s:~/桌面/LD\_PRELOAD$ gcc myverifypasswd.c -L. -lmystrcmp -o myverifypasswd

n6@X240s:~/桌面/LD\_PRELOAD$ export LD\_LIBRARY\_PATH=/home/n6/桌面/LD\_PRELOAD

n6@X240s:~/桌面/LD\_PRELOAD$ ldd myverifypasswd

linux-vdso.so.1 => (0x00007ffedc1bd000)

libmystrcmp.so => /home/n6/桌面/LD\_PRELOAD/libmystrcmp.so (0x00007eff99a7f000)

libc.so.6 => /lib/x86\_64-linux-gnu/libc.so.6 (0x00007eff99698000)

/lib64/ld-linux-x86-64.so.2 (0x000056269aba4000)

n6@X240s:~/桌面/LD\_PRELOAD$ ./myverifypasswd

usage: ./myverifypasswd <password>

n6@X240s:~/桌面/LD\_PRELOAD$ ./myverifypasswd abc

Invalid Password!

n6@X240s:~/桌面/LD\_PRELOAD$ ./myverifypasswd password

Correct Password!

**五、替换库代码**

// myhack.c

#include <stdio.h>

#include <string.h>

int mystrcmp(const char \*s1, const char \*s2)

{

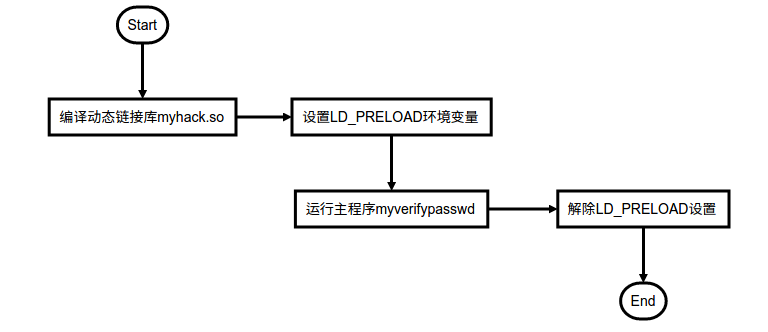
printf("hack function invoked. s1=<%s> s2=<%s>\n", s1, s2);

// always return 0, which means s1 equals to s2--总是相等

return 0;

}

**六、替换并测试运行**

* 替换流程图  
  
* 编译、设置指令
* gcc myhack.c -fPIC -shared -o myhack.so #编译替换库
* export LD\_PRELOAD="./myhack.so" #设置LD\_PRELOAD环境变量,库中的同名函数在程序运行时优先调用

./myverifypasswd #运行主程序

* 终端运行结果
* n6@X240s:~/桌面/LD\_PRELOAD$ gcc myhack.c -fPIC -shared -o myhack.so
* n6@X240s:~/桌面/LD\_PRELOAD$ export LD\_PRELOAD="./myhack.so"
* n6@X240s:~/桌面/LD\_PRELOAD$ ./myverifypasswd
* usage: ./myverifypasswd <password>
* n6@X240s:~/桌面/LD\_PRELOAD$ ./myverifypasswd abc
* hack function invoked. s1=<password> s2=<abc>
* Correct Password!
* n6@X240s:~/桌面/LD\_PRELOAD$ ./myverifypasswd password
* hack function invoked. s1=<password> s2=<password>
* Correct Password!

n6@X240s:~/桌面/LD\_PRELOAD$ unset LD\_PRELOAD

**七、LD\_PRELOAD运用总结**

1. 定义与目标函数完全一样的函数，包括名称、变量及类型、返回值及类型等
2. 将包含替换函数的源码编译为动态链接库
3. 通过命令 export LD\_PRELOAD="库文件路径"，设置要优先替换动态链接库
4. 如果找不替换库，可以通过 export LD\_LIBRARY\_PATH=库文件所在目录路径，设置系统查找库的目录
5. 替换结束，要还原函数调用关系，用命令unset LD\_PRELOAD 解除
6. 想查询依赖关系，可以用ldd 程序名称