

# [TOOL] Kfence, Kernel Protection against Basic Exploitation Techniques

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-08/0079.html>

---

**From:** SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

**Date:** 08/26/03

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 26 Aug 2003 16:06:45 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

-----

Kfence, Kernel Protection against Basic Exploitation Techniques

---

## DETAILS

Kfence provides kernel protection against basic exploitation techniques, including stack and heap overflows and format string exploit, by patching /dev/kmem and redirecting system\_call to test if the EIP of the caller is in the wrong memory region.

Tool code:

```
/*
 * kfence 1.2
 * added .bss exec protection
 * modified extraction of struct distances
 * you need to have a valid System.map on your box
 * tested on 2.2.16 2.4.7 2.4.18 2.4.19 2.4.10
 * Coded by ins1der
 * 2003
 * trixterjack@yahoo.com
 */
```

```
#define _GNU_SOURCE
```

```
#include <stdio.h>
```

## Securiteam: [TOOL] Kfence, Kernel Protection against Basic Exploitation Techniques

```
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <asm/unistd.h>
#define ulong unsigned long
#define systemmap "/boot/System.map"
struct _idt
{
    unsigned short offset_low,segment_sel;
    unsigned char reserved,flags;
    unsigned short offset_high;
};

char shellc[]=
    /*start:*/
"\x83\xf8\x21" /*cmp $0x21,%eax*/
"\x74\x32" /*je exitme*/
"\x55" /*push %ebp*/
"\x89\xe5" /*mov %esp,%ebp*/
"\x60" /*pusha*/
"\x8b\x55\x04" /*mov 0x4(%ebp),%edx*/
"\x81\xfa\xfe\xff\xff\xbf" /*cmp $0xbfffffff,%edx*/
"\x77\x08" /*ja test1*/
"\x81\xfa\x00\x00\x00\xbf" /*cmp $0xbf000000,%edx*/
"\x77\x1b" /*ja exitme*/
/*test1:*/
"\xb8\x00\xe0\xff\xff" /*mov $0xffffe000,%eax*/
"\x21\xe0" /*and %esp,%eax*/
"\x8b\x80\x00\x00\x00\x00" /*mov 0x0(%eax),%eax*/
"\x3b\x50\x00" /*cmp 0x0(%eax),%edx*/
"\x76\x05" /*jbe endme*/
"\x3b\x50\x00" /*cmp 0x0(%eax),%edx*/
"\x72\x04" /*jb exitme*/
/*endme:*/
"\x61" /*popa*/
"\x5d" /*pop %ebp*/
"\xeb\x06" /*jmp realend*/
/*exitme:*/
"\x31\xc0" /*xor %eax,%eax*/
"\xfe\xc0" /*inc %al*/
"\xcd\x80" /*int $0x80*/
/*realend:*/
;
int verbose=0;
/* read or write to /dev/kmem */

int kmemrw(void *rwbuf, ulong offset, size_t size, int type)
{
    int f;
    int ret;
    f=open("/dev/kmem",O_RDWR);
```

```

if (f<0)
{
fprintf(stderr,"unable to open /dev/kmem in rw mode\n");
_exit(0);
}
lseek(f,offset,SEEK_SET);
if (type==0)
ret=read(f,rwbuf,size);
else
ret=write(f,rwbuf,size);
close(f);
return ret;
}

void usage(char *argv)
{
fprintf(stderr,"***\nkfence \nins1der 2003
(trixterjack@yahoo.com)\n***\n");
fprintf(stderr,"Usage : %s command [v]\n",argv);
fprintf(stderr,"Commands:\n");
fprintf(stderr," r remove the kernel patch\n");
fprintf(stderr," i install kfence \n");
_exit(0);
}

int main(int argc,char *argv[])
{

unsigned char idtr[6],shell[100],start_data;
struct _idt idt;
ulong olduname,sct,len,system_call,readbytes,mm_dist=0;
ulong load_aout;
char *p;

if ((argc<2)||(argv[1][0]!='r'&&argv[1][0]!='i')) usage(argv[0]);
if (argc==3)
{
if (argv[2][0]!='v') usage(argv[0]);
else verbose=1;
}
if (argv[1][0]=='i')
printf("Starting kfence 1.2\n");
else
printf("Removing kfence 1.2\n");

{
FILE *f;
char buf[256];
f=fopen(systemmap,"r");
if (f==NULL)
{

```

## Securiteam: [TOOL] Kfence, Kernel Protection against Basic Exploitation Techniques

```
printf("System.map path not good maybe?\n");
_exit(0);
}

do
{
fgets(buf,sizeof(buf),f);
if (memmem(buf,256,"load_aout_interp",16)) break;
}
while (!feof(f));

if (feof(f))
{
printf("load_aout_interp not found\n");
fclose(f);
_exit(0);
}

sscanf(buf,"%x",(int*)&load_aout);

fclose(f);
}

/*get the interrupt descriptor table*/

asm("sidt %0" : "=m" (idtr));

/*get the address of system_call*/

kmemrw(&idt,*((ulong*)&idtr[2])+8*0x80,sizeof(idt),0);
system_call=(idt.offset_high<<16)|idt.offset_low;

if (verbose)
printf("# system_call at 0x%x\n",(int)system_call);

/*read the first 4 bytes of system_call*/

kmemrw(&readbytes,system_call,4,0);

if (argv[1][0]=='r')
{
if (readbytes==0x1e06fc50)

/*if the first bytes of system_call are normal kfence is not
installed*/

printf("kfence is not up. nothing to remove.\n");
```

```

else

{

/*remove kfence by writing the usual bytes back in system_call*/

kmemrw("\x50\xfc\x06\x1e\x50\x55",system_call,6,1);
printf("Done.\n");

}

return 0;

}

if (readbytes!=0x1e06fc50)
{

fprintf(stderr,"Something is wrong with system_call\n");
fprintf(stderr,"kfence probably installed\n");
fprintf(stderr,"Exiting.\n");
_exit(0);

}

if (verbose)
printf("# load_aout_interp at 0x%x\n",(int)load_aout);

/*get sys_call_table*/

kmemrw(shell,system_call,100,0);
p=(char*)memmem(shell,100,"\xff\x14\x85",3);
if (!p)
{
fprintf(stderr,"sys_call_table not found!\n");
return 0;
}
p+=3;
sct=(ulong*)p;
if (verbose)
printf("# sys_call_table 0x%x\n",(int)sct);

/*get the address of sys_olduname*/

kmemrw(&olduname,sct+4*__NR_oldolduname,4,0);
if (verbose)
printf("# olduname at 0x%x\n",(int)olduname);

/*get the distances from load_aout_interp bling bling! :\*/

```

```

p=(char*)kmemrw(shell,load_aout,70,0);
p=(char*)memmem(shell,70,"\x00\xe0\xff\xff",4);
if (!p)
{
    fprintf(stderr,"couldn't get needed structures!\n");
    return 0;
}

p--;
{
    unsigned char i,j,b,c;
    i=*p;
    i-=0xb8;
for (;;)
{
    p=(char*)memchr(p,'\x8b',70-((ulong)p-(ulong)shell));
    if (!p)
    {
        fprintf(stderr,"couldn't get needed structures!\n");
        return 0;
    }
    p++;
    b=*p;
    b-=0x40;

    if (b<=0x3f)

    if (b%0x8==i)
    {
        p++;
        mm_dist=*p;
        for (;;)
        {
            j=b/0x8;
            p=(char*)memchr(p,'\x89',70-((ulong)p-(ulong)shell));
            if (!p)
            {
                fprintf(stderr,"couldn't get needed structures!\n");
                return 0;
            }

            p++;
            c=*p;
            c-=0x40;
            if (c%0x8==j)
            {
                p++;
                start_data=*p+4;
                break;
            }
        }
    }
}

```

```

    }
}
break;
}

if((b>=0x40)&&(b<=0x80))
{
b-=0x40;
if (b%0x8==i)
{
p++;
mm_dist=*(int*)p;
p+=6;
for (;;)
{
j=b/0x8;
p=(char*)memchr(p,'\x89',70-((ulong)p-(ulong)shell));
if (!p)
{
fprintf(stderr,"couldn't get needed
structures!\n");
return 0;
}

p++;
c=*p;
c-=0x40;
if (c%0x8==j)
{
p++;
start_data=*p+4;
break;
}
}
break;
}

}

}

if (verbose)
{
printf("# mm distance in task_struct = 0x%x\n",(int)mm_dist);
printf("# start_data distance in mm_struct = 0x%x\n",start_data);
printf("##Sleeping for 2 seconds.\n");
}
sleep(1);
printf(".");

```

## Securiteam: [TOOL] Kfence, Kernel Protection against Basic Exploitation Techniques

```
fflush(stdout);
sleep(1);
printf(".\n");

/*prepare our code*/

len=sizeof(shellc)-1;
shellc[2]=__NR_oldolduname;
*(int*)(shellc+37)=mm_dist;
shellc[43]=start_data;
shellc[48]=start_data+0xc;
memcpy(shell,shellc,len);
kmemrw(shell+len,system_call,6,0);
*(char*)(shell+len+6)=0x68;
*(int*)(shell+len+7)=(int)(system_call+6);
*(char*)(shell+len+11)=0xc3;

/*write the code in sys_olduname*/

kmemrw(shell,olduname,len+12,1);
shell[0]=0x68;
*(int*)(shell+1)=olduname;
shell[5]=0xc3;

/*write a push ret at the begining system_call to jump to
sys_olduname*/

kmemrw(shell,system_call,6,1);

printf("Done.\n");

return 1;
}
```

### ADDITIONAL INFORMATION

The information has been provided by <<mailto:trixterjack@yahoo.com>>  
ins1der.

=====

This bulletin is sent to members of the SecuriTeam mailing list.  
To unsubscribe from the list, send mail with an empty subject line and body to:  
[list-unsubscribe@securiteam.com](mailto:list-unsubscribe@securiteam.com)  
In order to subscribe to the mailing list, simply forward this email to: [list-subscribe@securiteam.com](mailto:list-subscribe@securiteam.com)

=====  
=====

### DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

Securiteam: [TOOL] Kfence, Kernel Protection against Basic Exploitation Techniques

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.