

Re: [PATCH] Avoid buffer overflows in get_user_pages()

Re: [PATCH] Avoid buffer overflows in get_user_pages()

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-02/msg05515.html>

- *From:* Nick Piggin <nickpiggin@xxxxxxxxxxxxx>
 - *Date:* Tue, 12 Feb 2008 16:56:09 +1100
-

On Tuesday 12 February 2008 14:16, Robert Hancock wrote:

Nick Piggin wrote:

On Tuesday 12 February 2008 10:17, Jonathan Corbet wrote:

Avoid buffer overflows in get_user_pages()

So I spent a while pounding my head against my monitor trying to figure out the vmsplice() vulnerability – how could a failure to check for *read* access turn into a root exploit? It turns out that it's a buffer overflow problem which is made easy by the way get_user_pages() is coded.

In particular, "len" is a signed int, and it is only checked at the *end* of a do { } while() loop. So, if it is passed in as zero, the loop will execute once and decrement len to -1. At that point, the loop will proceed until the next invalid address is found; in the process, it will likely overflow the pages array passed in to get_user_pages().

I think that, if get_user_pages() has been asked to grab zero pages, that's what it should do. Thus this patch; it is, among other things, enough to block the (already fixed) root exploit and any others which might be lurking in similar code. I also think that the number of pages should be unsigned, but changing the prototype of this

Re: [PATCH] Avoid buffer overflows in get_user_pages()

function probably
requires some more careful review.

Signed-off-by: Jonathan Corbet <corbet@xxxxxxx>

```
diff --git a/mm/memory.c b/mm/memory.c
index e5628a5..7f50fd8 100644
--- a/mm/memory.c
+++ b/mm/memory.c
@@ -989,6 +989,8 @@ int get_user_pages(struct
task_struct *tsk, struct
mm_struct *mm, int i;
unsigned int vm_flags;

+ if (len <= 0)
+ return 0;
```

BUG_ON()?

Well, not if the code involved in the exploit can pass a zero value,

Which is a bug, and you want to catch it.

otherwise it's just turning it into a DoS..

If it is due to a security bug, then the fix is to fix the point
where the kernel starts trusting an untrusted value. Not to hide
the bug like this. Arguably, a BUG_ON is better in the case of a
security hole because you want to halt the process as soon as you
de