Subject: Re: [PATCH] IA64,sparc: local DoS with corrupted ELFs
Posted by Kirill Korotaev on Thu, 07 Sep 2006 10:14:02 GMT
View Forum Message <> Reply to Message

> On Mon, 4 Sep 2006, Kirill Korotaev wrote:
>>this patch is already commited into -stable 2.6.17.y tree.
>
>
> I don't like it. Apparently the patch was bad, and broken on MIPS and
> parisc, and it was applied to the stable tree without being in the
> standard tree.
it is locally exploitable DoS IMHO.
I'm not sure what is the policy about security fixes, since Tony Luck didn't
show much interest in this fix and I kept pushing it myself.
Probably I should have published an exploit to make it moving direct way :)

And I really sorry for patch sent being broken a bit :/

> If MIPS and parisc don't matter for the stable tree (very possible - there
> are no big commercial distributions for them), then dammit, neither should
> ia64 and sparc (there are no big commercial distros for them either).
> Either way, it seems this didn't happen the way it should have.
>
> The proper fix would _seem_ to have the whole
>
> #ifndef arch_mmap_check
> #define arch_mmap_check(addr, len, flags) (0)
> #endif
>
> in the only file that actually _uses_ this, namely mm/mmap.c. Rather than
> pollute lots of architecture-specific files with this macro that nobody
> really is interested in except for ia64 and sparc.

Does the patch below looks better?
(compile tested on IA64)

```
diff --git a/arch/ia64/kernel/sys_ia64.c b/arch/ia64/kernel/sys_ia64.c
index 40722d8..5b190c9 100644
--- a/arch/ia64/kernel/sys_ia64.c
+++ b/arch/ia64/kernel/sys_ia64.c
@@ -163,10 +163,25 @@ sys_pipe (void)
  return retval;
 }

+int ia64_mmap_check(unsigned long addr, unsigned long len,
+  unsigned long flags)
```

```
+{
+ unsigned long roff;
+
+ /*
+  * Don't permit mappings into unmapped space, the virtual page table
+  * of a region, or across a region boundary.  Note: RGN_MAP_LIMIT is
+  * equal to 2^n-PAGE_SIZE (for some integer n <= 61) and len > 0.
+  */
+ roff = REGION_OFFSET(addr);
+ if ((len > RGN_MAP_LIMIT) || (roff > (RGN_MAP_LIMIT - len)))
+  return -EINVAL;
+ return 0;
+}
+
 static inline unsigned long
 do_mmap2 (unsigned long addr, unsigned long len, int prot, int flags, int fd, unsigned long pgoff)
 {
- unsigned long roff;
  struct file *file = NULL;

  flags &= ~(MAP_EXECUTABLE | MAP_DENYWRITE);
@@ -188,17 +203,6 @@ do_mmap2 (unsigned long addr, unsigned l
   goto out;
  }

- /*
-  * Don't permit mappings into unmapped space, the virtual page table of a region,
-  * or across a region boundary.  Note: RGN_MAP_LIMIT is equal to 2^n-PAGE_SIZE
-  * (for some integer n <= 61) and len > 0.
-  */
- roff = REGION_OFFSET(addr);
- if ((len > RGN_MAP_LIMIT) || (roff > (RGN_MAP_LIMIT - len))) {
-  addr = -EINVAL;
-  goto out;
- }
-
  down_write(&current->mm->mmap_sem);
  addr = do_mmap_pgoff(file, addr, len, prot, flags, pgoff);
  up_write(&current->mm->mmap_sem);
diff --git a/arch/sparc/kernel/sys_sparc.c b/arch/sparc/kernel/sys_sparc.c
index a41c8a5..94ff58c 100644
--- a/arch/sparc/kernel/sys_sparc.c
+++ b/arch/sparc/kernel/sys_sparc.c
@@ -219,6 +219,21 @@ out:
  return err;
 }

+int sparc_mmap_check(unsigned long addr, unsigned long len, unsigned long flags)
```

```
+{
+ if (ARCH_SUN4C_SUN4 &&
+     (len > 0x20000000 ||
+      ((flags & MAP_FIXED) &&
+       addr < 0xe0000000 && addr + len > 0x20000000)))
+  return -EINVAL;
+
+ /* See asm-sparc/uaccess.h */
+ if (len > TASK_SIZE - PAGE_SIZE || addr + len > TASK_SIZE - PAGE_SIZE)
+  return -EINVAL;
+
+ return 0;
+}
+
 /* Linux version of mmap */
 static unsigned long do_mmap2(unsigned long addr, unsigned long len,
  unsigned long prot, unsigned long flags, unsigned long fd,
@@ -233,25 +248,13 @@ static unsigned long do_mmap2(unsigned l
    goto out;
  }

- retval = -EINVAL;
  len = PAGE_ALIGN(len);
- if (ARCH_SUN4C_SUN4 &&
-     (len > 0x20000000 ||
-      ((flags & MAP_FIXED) &&
-       addr < 0xe0000000 && addr + len > 0x20000000)))
-  goto out_putf;
-
- /* See asm-sparc/uaccess.h */
- if (len > TASK_SIZE - PAGE_SIZE || addr + len > TASK_SIZE - PAGE_SIZE)
-  goto out_putf;
-
  flags &= ~(MAP_EXECUTABLE | MAP_DENYWRITE);

  down_write(&current->mm->mmap_sem);
  retval = do_mmap_pgoff(file, addr, len, prot, flags, pgoff);
  up_write(&current->mm->mmap_sem);

-out_putf:
  if (file)
   fput(file);
 out:
diff --git a/arch/sparc64/kernel/sys_sparc.c b/arch/sparc64/kernel/sys_sparc.c
index 054d0ab..bf5f14e 100644
--- a/arch/sparc64/kernel/sys_sparc.c
+++ b/arch/sparc64/kernel/sys_sparc.c
@@ -548,6 +548,26 @@ asmlinkage long sparc64_personality(unsi
```

```
  return ret;
 }

+int sparc64_mmap_check(unsigned long addr, unsigned long len,
+  unsigned long flags)
+{
+ if (test_thread_flag(TIF_32BIT)) {
+  if (len >= STACK_TOP32)
+   return -EINVAL;
+
+  if ((flags & MAP_FIXED) && addr > STACK_TOP32 - len)
+   return -EINVAL;
+ } else {
+  if (len >= VA_EXCLUDE_START)
+   return -EINVAL;
+
+  if ((flags & MAP_FIXED) && invalid_64bit_range(addr, len))
+   return -EINVAL;
+ }
+
+ return 0;
+}
+
 /* Linux version of mmap */
 asmlinkage unsigned long sys_mmap(unsigned long addr, unsigned long len,
  unsigned long prot, unsigned long flags, unsigned long fd,
@@ -563,27 +583,11 @@ asmlinkage unsigned long sys_mmap(unsign
  }
  flags &= ~(MAP_EXECUTABLE | MAP_DENYWRITE);
  len = PAGE_ALIGN(len);
- retval = -EINVAL;
-
- if (test_thread_flag(TIF_32BIT)) {
-  if (len >= STACK_TOP32)
-   goto out_putf;
-
-  if ((flags & MAP_FIXED) && addr > STACK_TOP32 - len)
-   goto out_putf;
- } else {
-  if (len >= VA_EXCLUDE_START)
-   goto out_putf;
-
-  if ((flags & MAP_FIXED) && invalid_64bit_range(addr, len))
-   goto out_putf;
- }

  down_write(&current->mm->mmap_sem);
  retval = do_mmap(file, addr, len, prot, flags, off);
```

```
  up_write(&current->mm->mmap_sem);

-out_putf:
  if (file)
   fput(file);
 out:
diff --git a/include/asm-generic/mman.h b/include/asm-generic/mman.h
diff --git a/include/asm-ia64/mman.h b/include/asm-ia64/mman.h
index 6ba179f..936fa9c 100644
--- a/include/asm-ia64/mman.h
+++ b/include/asm-ia64/mman.h
@@ -22,4 +22,12 @@ #define MAP_NONBLOCK 0x10000  /* do not
 #define MCL_CURRENT 1  /* lock all current mappings */
 #define MCL_FUTURE 2  /* lock all future mappings */

+#ifdef __KERNEL__
+#ifndef __ASSEMBLY__
+#define arch_mmap_check ia64_mmap_check
+int ia64_mmap_check(unsigned long addr, unsigned long len,
+  unsigned long flags);
+#endif
+#endif
+
 #endif /* _ASM_IA64_MMAN_H */
diff --git a/include/asm-sparc/mman.h b/include/asm-sparc/mman.h
index 88d1886..b7dc40b 100644
--- a/include/asm-sparc/mman.h
+++ b/include/asm-sparc/mman.h
@@ -35,4 +35,12 @@ #define MC_UNLOCKAS     6  /* Unlock ent

 #define MADV_FREE 0x5  /* (Solaris) contents can be freed */

+#ifdef __KERNEL__
+#ifndef __ASSEMBLY__
+#define arch_mmap_check sparc_mmap_check
+int sparc_mmap_check(unsigned long addr, unsigned long len,
+  unsigned long flags);
+#endif
+#endif
+
 #endif /* __SPARC_MMAN_H__ */
diff --git a/include/asm-sparc64/mman.h b/include/asm-sparc64/mman.h
index 6fd878e..8cc1860 100644
--- a/include/asm-sparc64/mman.h
+++ b/include/asm-sparc64/mman.h
@@ -35,4 +35,12 @@ #define MC_UNLOCKAS     6  /* Unlock ent

 #define MADV_FREE 0x5  /* (Solaris) contents can be freed */
```

```
+#ifdef __KERNEL__
+#ifndef __ASSEMBLY__
+#define arch_mmap_check sparc64_mmap_check
+int sparc64_mmap_check(unsigned long addr, unsigned long len,
+  unsigned long flags);
+#endif
+#endif
+
 #endif /* __SPARC64_MMAN_H__ */
diff --git a/mm/mmap.c b/mm/mmap.c
index c1868ec..e66a0b5 100644
--- a/mm/mmap.c
+++ b/mm/mmap.c
@@ -30,6 +30,10 @@ #include <asm/uaccess.h>
 #include <asm/cacheflush.h>
 #include <asm/tlb.h>

+#ifndef arch_mmap_check
+#define arch_mmap_check(addr, len, flags) (0)
+#endif
+
 static void unmap_region(struct mm_struct *mm,
   struct vm_area_struct *vma, struct vm_area_struct *prev,
   unsigned long start, unsigned long end);
@@ -913,6 +917,10 @@ unsigned long do_mmap_pgoff(struct file
  if (!len)
   return -EINVAL;

+ error = arch_mmap_check(addr, len, flags);
+ if (error)
+  return error;
+
  /* Careful about overflows.. */
  len = PAGE_ALIGN(len);
  if (!len || len > TASK_SIZE)
@@ -1859,6 +1867,7 @@ unsigned long do_brk(unsigned long addr,
  unsigned long flags;
  struct rb_node ** rb_link, * rb_parent;
  pgoff_t pgoff = addr >> PAGE_SHIFT;
+ int error;

  len = PAGE_ALIGN(len);
  if (!len)
@@ -1867,6 +1876,12 @@ unsigned long do_brk(unsigned long addr,
  if ((addr + len) > TASK_SIZE || (addr + len) < addr)
   return -EINVAL;
```

```
+ flags = VM_DATA_DEFAULT_FLAGS | VM_ACCOUNT | mm->def_flags;
+
+ error = arch_mmap_check(addr, len, flags);
+ if (error)
+  return error;
+
  /*
   * mlock MCL_FUTURE?
   */
@@ -1907,8 +1922,6 @@ unsigned long do_brk(unsigned long addr,
  if (security_vm_enough_memory(len >> PAGE_SHIFT))
   return -ENOMEM;

- flags = VM_DATA_DEFAULT_FLAGS | VM_ACCOUNT | mm->def_flags;
-
  /* Can we just expand an old private anonymous mapping? */
  if (vma_merge(mm, prev, addr, addr + len, flags,
      NULL, NULL, pgoff, NULL))
```