

[PATCH]: non-readable binaries – binfmt_misc 2.6.0-test4

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2003-08/8362.html>

From: Zach, Yoav (yoav.zach_at_intel.com)

Date: 08/31/03

Date: Mon, 1 Sep 2003 00:41:23 +0300
To: <akpm@osdl.org>, <torvalds@osdl.org>

The proposed patch solves a problem for interpreters that need to execute a non-readable file, which cannot be read in userland. To handle such cases the interpreter must have the kernel load the binary on its behalf. The proposed patch handles this case by telling binfmt_misc, by a special flag in the registration string, to open the binary for reading and pass its descriptor as argv[1], instead of passing the binary's path. Old behavior of binfmt_misc is kept for interpreters which do not specify this special flag. The patch is against linux-2.6.0-test4. A similar one was posted twice on the list, on Aug. 14 and 21, without significant response.

===== start patch

```
=====
diff -r -U 3 linux-2.6.0-test4/fs/binfmt_misc.c linux/fs/binfmt_misc.c
--- linux-2.6.0-test4/fs/binfmt_misc.c Sat Aug 23 02:54:23 2003
+++ linux/fs/binfmt_misc.c Mon Sep 1 00:17:01 2003
@@ -38,6 +38,8 @@
```

```
enum {Enabled, Magic};
#define MISC_FMT_PRESERVE_ARGV0 (1<<31)
+#define MISC_FMT_OPEN_BINARY (1<<30)
+
```

```
typedef struct {
    struct list_head list;
@@ -106,6 +108,10 @@
    char *iname_addr = iname;
    int retval;
```

```
+ int fd;
+ char fd_str[32];
+ char * fdsp = fd_str;
+
    retval = -ENOEXEC;
    if (!enabled)
```

```

        goto _ret;
@@ -119,16 +125,41 @@
        if (!fmt)
            goto _ret;

- allow_write_access(bprm->file);
- fput(bprm->file);
- bprm->file = NULL;
+ if (fmt->flags & MISC_FMT_OPEN_BINARY) {
+ /* if the binary should be opened on behalf of the
+ * interpreter than keep it open and assign it a
+ descriptor */
+ fd = get_unused_fd ();
+ if (fd < 0) {
+ allow_write_access(bprm->file);
+ fput(bprm->file);
+ bprm->file = NULL;
+ retval = fd;
+ goto _ret;
+ } else {
+ fd_install (fd, bprm->file);
+ sprintf (fd_str, "/dev/fd/%d", fd);
+ }
+ } else {
+ allow_write_access(bprm->file);
+ fput(bprm->file);
+ bprm->file = NULL;
+ }
+

        /* Build args for interpreter */
        if (!(fmt->flags & MISC_FMT_PRESERVE_ARGV0)) {
            remove_arg_zero(bprm);
        }
- retval = copy_strings_kernel(1, &bprm->interp, bprm);
+
+ if (fmt->flags & MISC_FMT_OPEN_BINARY) {
+ /* make argv[1] be the file descriptor */
+ retval = copy_strings_kernel(1, &fdsp, bprm);
+ } else {
+ /* make argv[1] be the path to the file */
+ retval = copy_strings_kernel(1, &bprm->interp, bprm);
+ }
+
+ if (retval < 0) goto _ret;
+
+ bprm->argc++;
+ retval = copy_strings_kernel(1, &iname_addr, bprm);
+ if (retval < 0) goto _ret;
@@ -139,9 +170,18 @@
+ retval = PTR_ERR(file);
+ if (IS_ERR(file))

```

```
        goto _ret;
- bprm->file = file;

- retval = prepare_binprm(bprm);
+ /* get the file permissions and read its header */
+ if (fmt->flags & MISC_FMT_OPEN_BINARY) {
+   retval = prepare_binprm(bprm);
+   bprm->file = file;
+   memset(bprm->buf,0,BINPRM_BUF_SIZE);
+   retval =
kernel_read(bprm->file,0,bprm->buf,BINPRM_BUF_SIZE);
+ } else {
+   bprm->file = file;
+   retval = prepare_binprm(bprm);
+ }
+
+   if (retval >= 0)
+       retval = search_binary_handler(bprm, regs);
_ret:
@@ -296,6 +336,10 @@
+       p++;
+       e->flags |= MISC_FMT_PRESERVE_ARGV0;
+   }
+ if (*p == 'O') {
+   p++;
+   e->flags |= MISC_FMT_OPEN_BINARY;
+ }

+   if (*p == '\n')
+       p++;
===== end patch =====
```

Yoav Zach
Performance Tools Lab
Intel Corp.

–
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>