

Modifying file access time upon exec...

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2005-05/0099.html>

From: Ken Smith (*kensmith_at_cse.Buffalo.EDU*)

Date: 05/26/05

To: freebsd-arch@freebsd.org

Date: Thu, 26 May 2005 16:24:25 -0400

I started working on this many months ago and just as it seemed like we got to a point the people involved (mostly Bruce Evans...) were satisfied with the proposed solution I got ... sidetracked...

The issue is that the standards say a file's access time should be modified when it gets executed. We don't do that. This patch takes care of it by adding a new vnode flag that is used by the kernel to indicate that it has blessed an update of a file's access time. The flag gets set at the point the file get executed and the individual filesystems are expected to handle the details. Some don't store access times so it's a non-issue for them. UFS needs to handle it the same way as it would modifying the access time for a read - it shouldn't block, needs to be careful about whether a snapshot is in progress, bypasses most security checks, etc.

This patch is mostly Bruce's work. The final version of what he sent me had more changes in it that weren't directly related to fixing the atime issue that I still need to look over and understand better... :-)

Any thoughts before I commit it? The patch itself is pretty small. But given the sections of code it's mucking with combined with it adding a little 'nit' filesystem implementers should be aware of I wanted to run it by as many clueful eyes as possible before doing the final commit.

Thanks.

Index: sys/kern/kern_exec.c

```
=====
RCS file: /home/ncvs/src/sys/kern/kern_exec.c,v
retrieving revision 1.271
diff -u -r1.271 kern_exec.c
--- sys/kern/kern_exec.c 3 May 2005 16:24:59 -0000 1.271
+++ sys/kern/kern_exec.c 26 May 2005 19:11:04 -0000
@@ -284,7 +284,7 @@
     register_t *stack_base;
     int error, len, i;
```

freebsd-arch: Modifying file access time upon exec...

```
    struct image_params image_params, *imgp;
- struct vattr attr;
+ struct vattr atimeattr, attr;
    int (*img_first)(struct image_params *);
    struct pargs *oldargs = NULL, *newargs = NULL;
    struct sigacts *oldsigacts, *newsigacts;
@@ -695,6 +695,18 @@
    exec_setregs(td, imgp->entry_addr,
        (u_long)(uintptr_t)stack_base, imgp->ps_strings);

+ /*
+ * Here we should update the access time of the file. This must
+ * be implemented by the underlying filesystem in the same way as
+ * access timestamps for a VOP_READ() because we want to avoid
+ * blocking and/or I/O, and have not called vn_start_write().
+ */
+ if ((imgp->vp->v_mount->mnt_flag & (MNT_NOATIME | MNT_RDONLY)) == 0) {
+ VATTR_NULL(&atimeattr);
+ atimeattr.va_vaflags |= VA_EXECVE_ATIME;
+ (void)VOP_SETATTR(img->vp, &atimeattr, td->td_ucred, td);
+ }
+
done1:
    /*
    * Free any resources malloc'd earlier that we didn't use.
```

Index: sys/sys/vnode.h

```
=====
RCS file: /home/ncvs/src/sys/sys/vnode.h,v
retrieving revision 1.299
diff -u -r1.299 vnode.h
--- sys/sys/vnode.h 27 Apr 2005 09:18:10 -0000 1.299
+++ sys/sys/vnode.h 26 May 2005 15:23:37 -0000
@@ -284,6 +284,7 @@
    */
#define VA_UTIMES_NULL 0x01 /* utimes argument was NULL */
#define VA_EXCLUSIVE 0x02 /* exclusive create request */
+#define VA_EXECVE_ATIME 0x04 /* setting atime for execve */
```

```
    /*
    * Flags for ioflag. (high 16 bits used to ask for read-ahead and
Index: sys/ufs/ufs/ufs_vnops.c
```

```
=====
RCS file: /home/ncvs/src/sys/ufs/ufs/ufs_vnops.c,v
retrieving revision 1.269
diff -u -r1.269 ufs_vnops.c
--- sys/ufs/ufs/ufs_vnops.c 18 May 2005 22:18:21 -0000 1.269
+++ sys/ufs/ufs/ufs_vnops.c 26 May 2005 19:31:59 -0000
@@ -443,6 +443,20 @@
    ((int)vap->va_bytes != VNOVAL) || (vap->va_gen != VNOVAL)) {
        return (EINVAL);
    }
}
```

Modifying file access time upon exec...

freebsd-arch: Modifying file access time upon exec...

```
+ /*
+ * Update the file's access time when it has been executed. We are
+ * doing this here to specifically avoid some of the checks done
+ * below -- this operation is done by request of the kernel and
+ * should bypass some security checks. Things like read-only
+ * checks get handled by other levels (e.g., ffs_update()).
+ */
+ if (vap->va_vaflags & VA_EXECVE_ETIME) {
+ ip->i_flag |= IN_ACCESS;
+ return (0);
+ }
+ /*
+ * Go through the fields and update iff not VNOVAL.
+ */
+ if (vap->va_flags != VNOVAL) {
+     if (vp->v_mount->mnt_flag & MNT_RDONLY)
+         return (EROFS);
@@ -493,9 +507,6 @@
+     }
+     if (ip->i_flags & (IMMUTABLE | APPEND))
+         return (EPERM);
- /*
- * Go through the fields and update iff not VNOVAL.
- */
+ if (vap->va_uid != (uid_t)VNOVAL || vap->va_gid != (gid_t)VNOVAL) {
+     if (vp->v_mount->mnt_flag & MNT_RDONLY)
+         return (EROFS);
--
- From there to here, from here to      |      Ken Smith
there, funny things are everywhere.    |      kensmith@cse.buffalo.edu
- Theodore Geisel                       |
```

freebsd-arch@freebsd.org mailing list
<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>
To unsubscribe, send any mail to "freebsd-arch-unsubscribe@freebsd.org"