

Re: File System ACLs: Where to go from here in FreeBSD?

Source: <http://www.derkeiler.com/Mailing-Lists/FreeBSD-Security/2005-09/0022.html>

From: Charles Swiger (cswiger_at_mac.com)

Date: 09/21/05

To: Robert Watson <rwatson@freebsd.org>

Date: Tue, 20 Sep 2005 18:25:32 -0400

[...I guess freebsd–security is the mailing list other replies (Allen) are using...]

Hi, Robert—

This big an email may have frightened away the usual suspects, or perhaps the discussion about bumping library version numbers is stealing too much attention. :-)

Nevertheless, I'll toss a few comments into the ring...

On Sep 17, 2005, at 8:19 AM, Robert Watson wrote:

> *(1) We follow the POSIX.1e specification for file creation modes, in which the ACL and umask are combined with the cmode to generate a conservative ACL. This was the same as IRIX, but different from Solaris; Linux adopted the Solaris model. Since then, IRIX has (I believe) also switched to the Solaris model. Our model is "conservative" in that it will never offer broader rights on a file than the umask permits, but it turns out to be quite useful in some environments to allow the ACL on a directory to override the umask of a program writing files there.*

The first system I saw ACLs on was AFS, which used ACLs to extend and override the Unix mode bits in a more specific fashion. I believe this is what you refer to as the "Solaris model", but let me try to be more specific:

Given a file "foo", with 664 (rw–rw–r–) permissions, you could add a user–specific positive permission (user "adam" can write to the file), and this would apply even if adam was not a member of the

FreeBSD–Security: Re: File System ACLs: Where to go from here in FreeBSD?

group associated with the file. Likewise, you could add a negative permission, such as "no access" for "betty", which would mean betty could not read/write/list/etc the file, even if betty was a member of the group which owns the file "foo".

Likewise, you could set up ACLs for groups, where a specific ACL would extend and override the Unix mode bits. If foo was group–owned by staff, you could define a negative ACL for "students" like "no write", and people who were in the students group would not have write access, even if those users were also part of the staff group.

[This was actually a pretty common case, as grad students were employed as TA's or as junior sysadmins keeping the computing clusters going.]

Evaluation order was apply the Unix mode bits as appropriate considering the uid/gid of the process/user accessing the resource, then apply group–based rules, then apply per–user rules, to end up with the final set of access rights.

> *Option (b) is an interesting new choice as compared to 1999, when
> NTFS ACLs were in the distinct minority in terms of the syntax and
> semantics they offered. However, they become much more appealing
> if we consider that there appears to be a much clearer mapping from
> NTFS ACLs to NFSv4 ACLs than there is from POSIX.1e ACLs to NFSv4
> ACLs. And the fact that Mike Smith at Apple has taken the time to
> make it sit behind our library for the Darwin implementation on HFS
> +, etc, is also quite interesting. When I implemented the library,
> it was my hope that it would support that sort of thing, but we
> never actually tried :-).*

I remember Mike from the Darwin lists, he's a good guy (IIRC), and Apple has felt a strong need to implement filesystem and network sharing semantics which work well with Windows/NTFS/CIFS and with Samba.

While many people using FreeBSD have less need for such interoperability, it wouldn't hurt to do better, especially if FreeBSD can take advantage of the work already put into Darwin.

> *If we don't start considering a move to Darwin/NTFS ACLs, then we
> run into a problem when it comes to implementing NFSv4 ACLs: the
> mapping and behavior is rather poor and unclear. There's an ID on
> the topic, which I basically read as saying "This is all rather
> hard and rather non–ideal". Apple has identified that, for them,
> compatibility with NT (and possibly NFSv4?) is the way to go, and
> they may be right.*

NFS filesharing is much less important to Apple, frankly, and I would not expect them to be taking the lead on NFSv4.

FreeBSD–Security: Re: File System ACLs: Where to go from here in FreeBSD?

I'm not even sure NFSv4 is really on their radar map at all compared to providing an integration of Samba, OpenLDAP, Kerberos with decent GUI management tools in order to provide a reasonable replacement for a Windows domain controller or ADC.

> *On the other hand, the result is much reduced possibility of clean interoperability with Linux, Solaris, IRIX, and so on. So there's a definite trade-off.*

This concern probably matters more to FreeBSD than to MacOS X.

The interoperability issues with ACLs are minor compared with issues such as UFS being case-sensitive, whereas NTFS, CIFS/SMB, and HFS+ all being case-insensitive (but perhaps case-preserving). There's also the issue of Unicode support in filesystem pathnames, which provides a significant advantage to HFS+ for "normal people" [1] using anything outside of ASCII or ISO-8859-1.

--

-Chuck

[1]: Read this as somebody's grandmother from Russia or China or whatever that wants to name files in her native language. :-)

freebsd-security@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-security>

To unsubscribe, send any mail to "freebsd-security-unsubscribe@freebsd.org"