Junji's Blog Site

Just another WordPress.com weblog

ACLs on Samba

Posted on August 10, 2007 | 5 Comments

(—by Dustin Puryear)

Samba, the free and open-source Server Message Block/Common Internet File System (SMB/CIFS) file server for Linux and UNIX systems, is almost a household name these days. The reason is simple: Systems administrators can use Samba to leverage Linux as a reliable, high-performance file server. Samba has a lot of really cool features, including support for Windows NT domains and Active Directory (AD). By joining AD, a Samba-based file server has all the benefits of a Windows-based file server, minus the licensing cost.

Even though Samba can be just as feature rich as a Windows-based file server, many systems administrators don't configure a crucial element: ACL support. ACLs are useful because they let users be very specific about who can and can't see their documents, which is important in an enterprise setting. Windows users are very familiar with ACLs. Unless you're using Simple File Sharing in Windows XP, you can view files' ACLs by going to a directory's or file's Properties section and selecting the Security tab.

For the purposes of discussing Samba's support for ACLs, I assume that you already understand the basics of Windows ACLs, Linux, and Samba. For more information, see the Learning

system (e.g., SUSE, Debian, Slackware) for ACL testing. If you're using an entirely different type of UNIX (e.g., Solaris), refer to your vendor documentation for how to perform file-level ACL commands similar to the commands in this article. My example Samba server is joined to AD; for more information about AD integration, see the sidebar "ACE Hiccups."

Linux File Permissions

Most Linux file systems are configured by default to support only the basic UNIX file permission model. As Figure 1 shows, this model has three sets of permissions that apply to the owner (also known as *user*), group owner, and everyone else (known as *other*). In the figure, the owner has the permissions rwx (read, write, execute), the group owner has rw- (read, write), and other has r– (read).

This standard UNIX model generally works well. However, caveats do exist—the most important of which is that for complex permission requirements, administrators must do some creative thinking for how to define groups. In addition, only the UNIX administrator defines group membership. These limitations reduce the flexibility of UNIX permissions. Fortunately, all modern UNIX and Linux systems support ACLs, even if they aren't used all that often.

Linux ACLs

To support ACLs with Linux, you must first ensure you're using a file system with ACL support. Examples of file systems that support ACLs include ext3 and ReiserFS; I used ext3 for my examples.

When evaluating Linux ACLs, keep two things in mind. First, ACL support on most UNIX and Linux systems follows the POSIX ACL specification. Alas, this "specification" isn't technically a standard, but it's widely implemented. Second, POSIX ACLs don't have one-to-one mapping with Windows ACLs. Windows ACLs, or, more specifically, access control entries (ACEs), support a much wider range of features than POSIX ACLs. For example, Windows has a Take Ownership ACE that has no corresponding permission in UNIX. POSIX uses only UNIX's basic read, write, and execute permission model to specify each ACE within an ACL.

Now let's review how to enable and manage ACLs on your Linux server directly. Understanding and being able to directly administer Linux ACLs will help you tweak and debug permission issues for real-world Samba implementations.

Enabling ACLs

Although ACLs are supported for ext3, they often aren't enabled by default. For example, RHEL4 doesn't enable ACLs by default, but SUSE does. Even though Samba itself supports ACLs, it uses the underlying file system to implement this support. So, if ACL support isn't enabled on your file system, Samba can't support ACLs for your Windows users. You need to determine whether ACLs are enabled, and you need to know how to enable them if not.

The easiest way to determine whether ACLs are enabled is to run the mount command—which is used to attach a file system to the server—then visually determine whether the acl option is enabled. To do so, enter

```
# mount -t ext3
```

Rather than have the mount command print out all mounted file systems, use the -t option to print out only ext3-based file systems. The command output is

```
/dev/mapper/VolGroup00-LogVol00 on / type ext3
(rw)
/dev/sda1 on /boot type ext3 (rw)
/dev/mapper/VolGroup00-SambaVol on /samba type
ext3 (rw)
```

Notice in this example that the file system /samba, which is where we'll store Samba file shares, shows only (rw). Thus, ACLs aren't yet enabled on this file system.

The next step is to enable ACL support for /samba. First, you must dynamically enable ACL support on the file system. You can accomplish this task without a reboot by using the -remount option with the mount command, as follows:

```
# mount –o remount,acl /samba
```

ACL support is now enabled for /samba. Entering the following command

```
# mount -t ext3
```

gives you the following output

```
/dev/mapper/VolGroup00-LogVol00 on / type ext3
(rw)
/dev/sda1 on /boot type ext3 (rw)
/dev/mapper/VolGroup00-SambaVol on /samba type
ext3 (rw,acl)
```

This quick solution lets you enable ACL support without requiring you to reboot. However, this change will be lost when the server does reboot. To make the change permanent, you need to edit the /etc/fstab configuration file. On my example system, /etc/fstab defines /samba as follows:

```
/dev/mapper/VolGroup00-SambaVol /samba ext3
defaults 0 0
```

To permanently enable ACL support, add acl after the defaults option:

```
/dev/mapper/VolGroup00-SambaVol /samba ext3
defaults,acl 0 0
```

Now the /samba file system supports ACLs and will continue to do so even after the next server reboot.

Using ACLs

After you enable ACLs on the file system, you can start doing some of the real work. For example, you can modify ACLs on Linux. Technically, you can use ACLs via Samba without knowing

how to manage them directly on the Linux server. However, actually knowing how ACLs work in Linux will help you tweak and debug your Samba file servers.

Creating a dedicated workspace is helpful. For my example, I created /samba/share1:

```
# mkdir /samba/share1
# chgrp "Domain Users" /samba/share1
# chmod 775 /samba/share1
# cd /samba/share1
```

In this example, I created the directory /samba/share1, set the group owner to "Domain Users", and allowed both the directory owner and users in "Domain Users" to have full read, write, and execute access. I now have a workspace to do some actual testing. In the following example, I list a directory and file that I created in /samba/share1:

```
# ll
total 1
drwx——  3 dpuryear Domain Users 1024 Apr 17
15:48 dir1
-rwx——  1 dpuryear Domain Users    0 Apr 17
10:01 file1.txt
# getfacl file1.txt
# file: file1.txt
# owner: dpuryear
# group: Domain40Users
user::rwx
```

```
group::—
other::—
```

As mentioned previously, Linux ACLs simply extend the existing UNIX permission model. You can see this by comparing the output from ls and getfacl, where getfacl is the command-line tool for retrieving a file's or directory's ACL information:

```
# ll
total 1
drwx——  3 dpuryear Domain Users 1024 Apr 17
15:48 dir1
-rwx——  1 dpuryear Domain Users    0 Apr 17
10:01 file1.txt
# getfacl file1.txt
# file: file1.txt
# owner: dpuryear
# group: Domain40Users
user::rwx
group::—
other::—
```

This output shows that the files are owned by the Active Directory (AD) user dpuryear and that the group owner is Domain Users. In Linux, all files must have both an owner and group owner—unlike Windows, which can actually have a completely empty ACL.

Now, let's modify the ACL on file1 to accomplish some

common Windows tasks. First, let's give Jane and Bob read access to file1.txt:

```
# setfacl -m u:jane:r,u:bob:r file1.txt
```

Next, let's give read and write access to everyone in the Accounting group:

```
# setfacl -m g:Accounting:rw file1.txt
```

To see the effects of these changes, run getfacl again:

```
# getfacl file1.txt
# file: file1.txt
# owner: dpuryear
# group: Domain40Users
user::rwx
user:jane:r-
user:bob:r-
group::—
group:Accounting:rw-
mask::rw-
other::—
```

Windows administrators would typically see these permissions documented as follows:

```
Owner:F
Jane:R
```

```
Bob:R
Domain Users:N
Accounting: W
Everyone:N
```

Another important Windows concept that you need to understand is inheritance. The concept of inheritance is implemented in Samba, but a similar concept also exists in Linux ACLs, using "default" permissions on a directory. Specifically, when setting default permissions you specify the permissions that will be given to any file or directory created within the specified directory. This capability is very powerful. In the following example, I ensure that any files or directories created below dir1 have the necessary permissions:

```
# setfacl -m u:jane:r,u:bob:r,g:Accounting:rw
dir1
# setfacl -m
d:u:jane:r,d:u:bob:r,d:g:Accounting:rw dir1
# getfacl dir1
# file: dir1
# owner: dpuryear
# group: Domain40Users
user::rwx
user:jane:r-
user:bob:r-
group::—
group:Accounting:rw-
mask::rw-
```

```
other::-
default:user::rwx
default:user:jane:r-
default:user:bob:r-
default:group::-
default:group:Accounting:rw-
default:mask::rw-
default:other::-
```

Notice the two nearly identical setfacl commands. The first setfacl command sets the actual permissions on the directory, whereas the second setfacl command (i.e., the command that includes the d: option) sets the default permissions.

Samba and ACLs

The next step is to take a closer look at Samba's ACL support. As I noted earlier, Samba relies heavily on the underlying file system, so ACL support must be enabled on both the file system and on Samba itself. As of Samba 3.0, ACL support is enabled automatically (assuming that ACL support is also enabled on the file system). To explicitly enable ACL support on Samba, you can use the "nt acl support = yes" option in each share definition, as follows:

```
[share1]
   path = /samba/share1
   nt acl support = yes
   writeable = yes
```

You also need experience with troubleshooting ACLs. To practice troubleshooting, temporarily disable ACL support in share1 as follows:

```
[share1]
   path = /samba/share1
   nt acl support = no
   writeable = yes
```

As Figure 2 shows, if you then click the Properties view for a file in share1, you don't see a Security tab.

If you reenable NT ACL support in share1 but disable it on the file system, you'll see the Security tab, but only the owner, the group owner, and the Everyone group (which Samba maps to the "other" UNIX permission group) will be visible. If you try to add another ACE for the Accounting group, you'll receive an error message like the one in Figure 3, because Samba can't use the traditional UNIX permission model to store an additional ACE.

To enable ACL support on the file system, enter

```
# mount -o remount,acl /samba
```

To ensure that Samba supports ACLs, set the [share1] configuration in smb.conf to the following:

```
[share1]
   path = /samba/share1
   nt acl support = yes
   writeable = yes
```

Now, let's use the Windows Security tab to view the ACL for file1.txt, as Figure 4 shows. Notice that the ACLs we used setfacl to manually define (i.e., Bob and Jane) are now visible. In addition, you can now use the Security tab to modify ACLs as necessary.

Next, let's focus on how to handle ACLs for directories. In a previous section I demonstrated how to use setfacl to manually create ACLs. Now, we'll do this using the Windows interface only.

First, use Windows Explorer to create a directory in \\samba \share1\ named dir2. The Security tab for this folder will look similar to the one that Figure 5 shows.

Notice that the owner, dpuryear, has Special Permissions (which map to the UNIX permissions of rwx). Having Special Permissions might be confusing to Windows users, so Samba 3.0.20 and later support an "acl map full control = yes" parameter that shows rwx as FULL CONTROL. Alternatively, if you click FULL CONTROL for an ACE in older versions (e.g., to apply FULL CONTROL to dpuryear), then Samba will make some adjustments to the access and default permissions so that a Windows user sees FULL CONTROL in the Security tab.

Retire Your Windows File Servers

Many systems administrators don't realize that a Samba server, especially when fully integrated with AD, can truly serve as a drop-in replacement for a Windows file server. Samba's ACL support gives your users the full power of Windows ACLs, but at the price point of Linux and Samba. You've gotta love that.

Dustin Puryear (dustin@puryear-it.com) provides expertise in identity management, directory services, and Linux interoperability. He is the author of *Best Practices for Managing Linux and UNIX Servers* (*Windows IT Pro* eBooks).

Sidebar: ACE Hiccups

One problem with Samba ACL support is that listing users to use for access control entries (ACEs) within ACLs can be troublesome. Specifically, if you're using Samba in a standalone mode (i.e., configured with "user" security mode), Windows 2000 and Windows XP users might not be able to consistently list Samba users when configuring an ACL. However, when Samba is joined to a Windows NT domain or to Active Directory (AD), this problem doesn't occur. To follow the examples in this article, you need to configure Samba and winbind for integration into AD. Many Linux distributions supply tools to simply this task, such as Red Hat and CentOS' system-config-authentication. Alternatively, you can configure winbind by hand. Various tutorials, such as one on the Samba Web site (http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/winbind.html), explain this process.

Learning Path

Windows Access Control:

"Access Denied: Setting Permissions on Windows Server 2003 Shared Folders," InstantDoc ID 41280

"Logon Rights: The Heart of Windows Access Control," InstantDoc ID 46870

"Ups and Downs of AD Delegation," InstantDoc ID 15968

"Windows Server 2003: Secure By Default," InstantDoc ID 39808

Linux:

"Getting Started with Linux," http://www.linux.org/lessons/beginner/toc.html

Samba:

"Samba Basics," http://samba.org/samba/docs/man/Samba-Developers-Guide/pt02.html

Read more at techxworld.com

This entry was posted in Administration, Networking. Bookmark the permalink.

5 RESPONSES TO ACLS ON SAMBA

jim | August 20, 2007 at 4:40 pm | Reply

thanks for the great article! i used it to configure the ACL for our new samba server which is replacing our windows file servers. I've run into a problem the article doesn't address and might be useful for anyone security minded. after you join samba to AD, and enable ACL, i haven't found any way to RESTRICT regular users from changing the settings in the security tab from windows file explorer.
users adding themselves to sensitive folders is a problem and i haven't found anything to stop them. Dustin, do you know of a solution?
thanks again, jim

Brian | May 3, 2009 at 3:19 pm | Reply

This post is exactly what I've been looking for. Thanks for taking the time to put this out there. I have been digging through Samba docs for some time to try and get this working correctly, and for me the key was the acl on the file system. Once that was done, I didn't have to set all of the other directives in the smb.conf file, I set the group and bam! everything worked as advertised.

Thanks again.

Pingback: Sinkro.net · links for 2009-10-19

Tsquare | March 6, 2010 at 12:33 am | Reply

Great Blog!.....There's always something here to make me laugh...Keep doing what ya do

Daily File | October 2, 2010 at 4:46 am | Reply

wow..I used to be on the lookout for this and finally received it from this post. Thanks for making it easier.
