

Linux File Access Controls

Table of Contents

Discretionary and Mandatory Access Control	2
Linux Access Control Lists (ACL).....	3
Linux ACLs	4
Linux ACLs – setfacl.....	5
Linux ACLs – getfacl after setfacl	7
Linux ACLs – setfacl.....	8
Linux ACLs – getfacl.....	9
Linux ACLs – Least Privilege Concept.....	12
Discretionary Access Control (DAC).....	13
DAC – Downsides	14
Mandatory Access Control (MAC)	17
MAC – Advantages and Disadvantages.....	18
MAC – Implementations	19
Notices	20

Discretionary and Mandatory Access Control



Discretionary and Mandatory Access Control



**100 Jeff Arsenault: All right, so now we're going to cover discretionary and mandatory access controls and walk through those.

Linux Access Control Lists (ACL)

Can be implemented if a user wants more control over files than just standard permissions

Enabled but not typically used by default in many Linux distributions

- To turn functionality on, remount the needed mount point.

Can be enabled by modifying the mount point in `/etc/fstab`.

```
mount -o remount,acl /mount_point
```

Enabled by default on RHEL 5



**101 So in Linux we have another capability. So we talked about the regular type of permissions using `chmod` and `chown` and change group.

So now we're going to talk about-- Linux also supports access control lists. They're not enabled by default on most operating systems or most distributions of Linux. It is enabled by default on Red Hat Enterprise Linux 5 and above.

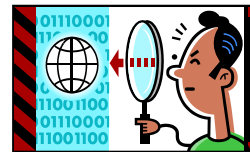
If you need to- if you're on another disk drive and you want to enable it, you'd have to enable it through the `fs` tab or through the `mount` command. So to do that we'd add the option `acl`, after the `remount`.

Linux ACLs

Can add permissions to other groups or users that typically do not have access to certain files

Example

Apache running content owned by a user. This typically should not be done since it is dangerous to allow Apache to run user or root content. We could allow group and other permissions, however this might also be a security risk because every user could look at everyone else's home directory.



**102 So when we add-- so if we want to have multiple users have permissions to the same file, the only way we can do it in the traditional Linux permissions scheme is to use groups. So if we don't want to do the extra groups, we can fine-grain using ACLs the kind of control that a user has. So we can say just a specific user has this specific kind of control on a file; instead of the more generic, broader way that the built in permission schemes allow.

So for instance, so Apache running content owned by a user; typically this shouldn't be done because it's dangerous. But if we want to say

Apache has permissions to a certain user's file to be able to read it, we can give the Apache user read access to that file and no other permissions to that file. Otherwise we'd have to give that entire group access; or we'd have to give everyone access to it. So this really lets us give it more control by per user and by permission.

Linux ACLs - setfacl

Linux ACLs – setfacl

Changes the permissions on your home directory for another group or user to access the content

```
setfacl -m u:apache:rx /home
```

Now, if we look at the permissions on the directory, we should see a “+” sign at the end of the standard permissions.

```
drwxr-x---+ 117 student student 4096 2013-04-02 15:54 student
```



**104 So setfacl-- this actually sets permissions. So in this one we're going to say: Set the facl for the /home directory so that the user, Apache, has read and execute.

Now we can tell if an ACL is set compared to the regular permissions because it'll have-- when you do a directory listing you'll see the little plus sign at the end of the permissions. So we have our usual permissions-- read, write, execute user; read and execute group; and everyone has none. But this plus indicates that there's extra permissions. So there's ACLs involved here. So we have to use a different command to see what those ACLs are.

Linux ACLs – getfacl after setfacl

Notice `#effective:---`

This means that Apache has no effective permissions set here.

- This should be entered when the `setfacl` command is entered.

To modify or remove permissions we use `setfacl` again.

```
setfacl -m u:apache:rx /home/student
setfacl -x u:apache /home/student
```

**105 And that's where `getfacl` comes in. So this will actually show us the ACL permissions that are on that file, as opposed to just the standard Linux permissions.

So here's another example. So the `-m` will modify the ACLs. So again, like in the previous example, this one we're going to do `home/student`. We're giving the user, Apache, re- execute. And then if we wanted to remove all the ACL permissions, we use a `setfacl -x`.

Now doing the `setfacl -x` we remove all the ACL permissions; but the standard Linux permissions still apply. So it's really the most restrictive permissions apply first.

Linux ACLs – setfacl

Changes the permissions on your home directory for another group or user to access the content

```
setfacl -m u:apache:rx /home
```

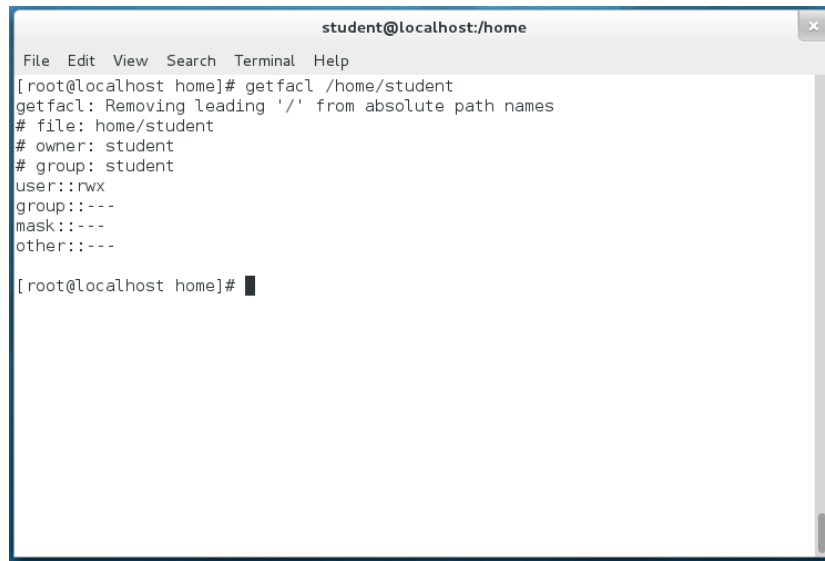
Now, if we look at the permissions on the directory, we should see a “+” sign at the end of the standard permissions.

```
drwxr-x---+ 117 student student 4096 2013-04-02 15:54 student
```



**104 So let me do a demo of this.

Linux ACLs – getfacl



```
student@localhost:/home
File Edit View Search Terminal Help
[root@localhost home]# getfacl /home/student
getfacl: Removing leading '/' from absolute path names
# file: home/student
# owner: student
# group: student
user::rwx
group::---
mask::---
other::---
```



**103 So it's easier to see than these screenshots.

So if we look-- so right now we are in the home directory for user virtual; that was the user I created for this instance. So that's in /home/virtual.

So if we want to see what the ACLs are on there right now, we type getfacl. So this means there's no-- this is showing the standard user group mask; this is the standard group permissions. This is not showing any ACLs yet.

So if we say-- shouldn't do this. I'm going to do root, just as an example; because I don't have Apache installed.

So now you see-- so the ACL there is the root- user root r -x. Then if I go remove it. With a command. That's what I thought. Now you see it's removed; and we're just back to the standard.

Student: And you said we could do this for both users and groups?

Jeff Arsenault: Yes, you can do it for the same- for the same mask. That's why I specified the u.

Student: Okay.

Jeff Arsenault: Now of course you still have to have the appropriate permissions to be able to sign the ACLs. So you have to have-- so here I'm as my user. So I can do it because I'm doing it to files that I own as the virtual user. Now I wouldn't be able to go set those ACLs on another user's file or on system files, unless I was root; or a member of a group that had permission to do that.

Student: So as an administrator can you restrict the use of that command to users? So that they cannot use that command. So in other words you don't want the users sharing their own stuff with other individuals.

Jeff Arsenault: Yes. So you'd set-- so you'd set the permissions on the set.

Student: On that executable?

Jeff Arsenault: On those executables themselves.

Student: Do that over.

Jeff Arsenault: Now in some of the advanced topics there's some-- like SELinux is a policy control. It's a MAC security control. We'll cover-- I'm going to mention that today but we'll cover it in more detail in another block. But you can assign- you can assign a very granular control of users that have permissions to execute which binaries; and what binaries are allowed to run is who's backup. So you can have very granular control on it.

But at the most basic level, you just don't let them run those commands; if you don't want them doing it on their own binaries.

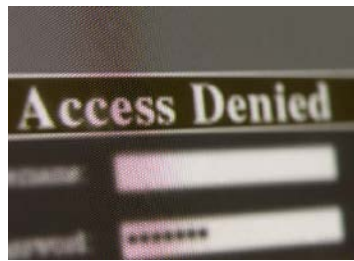
Linux ACLs – Least Privilege Concept

Important to understand

A security must

- Only giving users the least amount of access to do their

Provides granular controls to ensure that users or groups have the minimum amount of access needed



**107 So Least Privilege Concept; important to understand. So you want to make sure that they have the least amount of privileges possible and then add privileges as needed, so that they don't unnecessarily have control that they shouldn't.

And so ACLs are a way of doing that, to really give it more fine-grained control over what can run- who can run what; instead of a- instead of the broader permissions that the base Linux provides.

Discretionary Access Control (DAC)

Linux ACLs regarding files and permissions

What most people are familiar with and it works – mostly

A model that distrusts other users and isolates the damage that can be done to users

**108 So the ACLs regarding files and permissions. So this is what Discretionary Access Control is; and that's what the Linux ACLs are. A model that distrusts other users and isolates the damage that can be done to users; because we're saying users only have these specific privileges on these files.

But it's still kind of time consuming because you have to go and assign those permissions per file, per machine. It's not a centralized policy.

DAC – Downsides

Software is trusted.

All programs that a user has access to – the program also has access to the files.

Programs are not limited to what they are intended to do.

Programs could have vulnerabilities that could lead to a compromise of the system.

If a user has sudo or root privileges a system can be compromised by programs.



**109 So it doesn't care what the program is. It's just looking at the permissions of that program. So if software is trusted.

All programs that a user has access to, the program also has access to those files.

And programs themselves can still be vulnerable.

And then of course if they sudo or root. Sudo-- this is the first time it's mentioned; and we'll cover- we'll cover that again; we'll cover that in a later slide. But sudo is a way of

providing certain permissions to users to execute- to escalate their privileges.

So instead of actually logging in as root, I can use sudo; and it'll log my activities as root; and it'll log- it'll log my attempt to go into root also. So you have a log file; which is important.

But you can also use sudo to even limit it further; so you can define groups that can only run a certain set of binaries. And then so they sudo-- they run sudo in the command and if they have permissions to run that command then they'll be allowed to; otherwise it'll be disallowed.

So that's one of the ways you could probably-- that's another way you can limit the setfacts, limit that to a group. But you'd still- you'd have to assign those to a group in the regular base Linux, like we talked earlier.

Student: The way I've done it is I've just removed the executable completely from their path or from the directories that they had access to. So they wouldn't even get a chance to run any of these; anything I didn't want them to run. How does that work compared--

Jeff Arsenault: So if you remove it from their path they can still navigate to it; it's just not in their paths. So if they type if- if they type a full path from a command line they'll still be able to run it. It's just that they just typed it--

Student: Yes but I mean remove the executable completely; you know, like they have access to bin and sbin - give them their own flavor. And they think they're all-- they think they have everything. But you just- you just pulled out the stuff you didn't want them to have.

Jeff Arsenault: Well it depends on the binary. You have to do your research. But some of those might- some of the binaries might be necessary for the OS and other applications to run. So other applications, especially scripts, take advantage of a lot of the standard system binaries. So I'd be wary of moving anything out of bin or sbin, unless you're sure of all the dependencies on it.

Student: Okay.

Mandatory Access Control (MAC)

Access is centrally managed and mandatory controls are put in place.

A user may be given access to read a file but only by certain programs, and the reciprocal is true, a program can only read a set of user files.

MAC can work with DAC.



**110 So Mandatory Access Control, MAC, is access that's centrally managed and mandatory controls are put in place. So a user is given access to read a file but only by certain programs; and the reciprocal is true. We kind of just covered that.

So this is lot more centralized. So you can apply a policy to the entire machine; instead of these- as opposed to the DAC where we were assigning individuals ACLs to files.

And you can do both at the same time too. So if you really want to-- but you really have to plan out your permissions scheme. Because when

you start limiting a lot of permissions and you don't have a really good plan, you can start breaking stuff; and, you know, the people who need to access stuff, especially system files might not run properly anymore.

MAC – Advantages and Disadvantages

MAC – Advantages and Disadvantages

Advantages

- Distrusts everything
- Programs can be contained

Disadvantages

- Can be difficult to understand
- Access must be configured centrally
- Simple changes in the system may prevent access
 - Good thing for the security conscious
 - Time consuming if you are constantly changing components such as certificates



**111 So MAC automatically-- the advantages: It distrusts everything; programs can be contained.

Disadvantages: It's very complicated; especially when we start talking about SELinux in a later block, it's a lot more complicated on how to manage it. It's not as easy as these get and set facts.

Access must be configured centrally. So you'll have a giant- you'll have one policy that'll apply to the entire machine; and that policy will be applied on startup usually, instead of just dynamically like we were doing with the ACLs.

MAC - Implementations

MAC – Implementations

LIDS

- Kernel patch
- Can control every file
- Protect processes even from root

Tomoyo

- Automated policy configuration
- Friendly administration

SELinux

- Most widely used
- Developed by NSA
- Combines Role Based Access Control, Type Enforcement, and Multilevel security



**112 Here are some common implementations for MAC. And I'm just giving a brief overview here because we're going to go into these in more depth in a later block, to really focus on it. So we're just going to give a high level here.

So LIDS is a kernel patch; can control every file; protect processes even from root.

Tomoyo: An automated policy configuration with a more user-friendly administration.

SELinux comes with Red Hat Enterprise Linux by default. It's not enabled by default. You have to configure it. And we'll spend a good time on SELinux because that does come with Red Hat Enterprise. It was developed by NSA. It's the most widely used. Quite a few articles on it, on the Web.

Notices

Notices

© 2014 Carnegie Mellon University

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for the U.S. government purposes described below, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

This material was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The U.S. government's rights to use, modify, reproduce, release, perform, display, or disclose this material are restricted by the Rights in Technical Data-Noncommercial Items clauses (DFAR 252-227.7013 and DFAR 252-227.7013 Alternate I) contained in the above identified contract. Any reproduction of this material or portions thereof marked with this legend must also reproduce the disclaimers contained on this slide.

Although the rights granted by contract do not require course attendance to use this material for U.S. government purposes, the SEI recommends attendance to ensure proper understanding.

THE MATERIAL IS PROVIDED ON AN "AS IS" BASIS, AND CARNEGIE MELLON DISCLAIMS ANY AND ALL WARRANTIES, IMPLIED OR OTHERWISE (INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE, RESULTS OBTAINED FROM USE OF THE MATERIAL, MERCHANTABILITY, AND/OR NON-INFRINGEMENT).

CERT® is a registered mark owned by Carnegie Mellon University.