Access Control

Access Control Models

1. DAC
2. MAC
3. RBAC
4. ABAC

DAC

This is an access control policy that's determined by the owner. Every file out there and every folder is considered an object, and then you, the user, is considered an owner or whoever owns that file or folder. Now, the owner gets to decide who gets to read that. So, on this example, for instance, I have a folder and inside this folder it's owned by Jason. Jason has decided that I can read and write it because I'm Jason but my staff members can only read it, and everybody else on the network can only read it. This is why DAC is commonly used because you have very granular control to decide who has access to the things you've created. This works great if you want to be able to tell who can use which files and which folders, as the person who created it. Now, the problem with this is that you have to have two things being met. First, every object in a system has to have an owner. Nothing can be out there without an owner because if it had no owner, nobody would know who had the right permissions to it because the owner sets the permissions. And second, you need to make sure that each owner determines the access rights and permissions for each object. If I'm the owner of a file and I never set permissions on it, nobody is going to be able to read it.

MAC

MAC is going to be Mandatory Access Control. It's an access control policy where the computer system gets to decide who gets access to what objects. Now, how does it do that? Well, with Discretionary Access Control you, the owner, got to choose who got permissions. But in MAC, the computer's going to do that for you and it does this through data labels. So, with MAC, data labels create this trust level for all subjects and all objects. So, every person out there gets a label of if we have high trust, medium trust or low trust for them, and each data object gets a label, as well, high trust, medium trust, or low trust. Well, the most common use of Mandatory Access Control is in military context for high security systems. So, if you've seen a war movie at any time in your life, you've probably seen the words 'top secret' on some document. Well, there's really four levels of documentation inside the military context. They have the unclassified level. They have the confidential level. They have the secret level. And then, they have the top secret, which is the most secret. Now, each person gets a clearance level of what they're allowed to see. So, maybe the private only gets to see confidential information, and maybe the colonel, he gets to see the top secret information, and the captain only gets to see the secret. And so, each person gets a label associated with them and their clearance, and then the documents all get labeled with whatever they are, unclassified, confidential, secret, or top secret. If you want to access something, you need to not just meet the minimum level, but you also have to have what's called a need-to-know. So, for instance, let's say I have an army guy and a navy guy, and they both have a top secret clearance but it's about a navy operation. Then, maybe that army guy doesn't need to know about it, and he's not going to get access. Even though he has that clearance, right? So, these labels are very in-depth and they get very, very complicated. That's the idea here with MAC. Now, MAC is implemented through one of two ways. It can be Rule-based or it can be Lattice-based, and these are two different access control methods that are sub-methods of Mandatory Access Control.

RBAC

Role-Based Access Control is an access model that's controlled by the system, like MAC does, but instead, it focuses on a set of permissions vice an individual's permissions. So, we don't have to actually label each individual person on every single file. Instead, we can use roles for those people. So, the way I like to think about this is we create roles for each job function, and then we assign roles for each person's permissions to each object. So, let me give you an example. Role-Based Access Control is an access model that's controlled by the system, like MAC does, but instead, it focuses on a set of permissions vice an individual's permissions. So, we don't have to actually label each individual person on every single file. Instead, we can use roles for those people. So, the way I like to think about this is we create roles for each job function, and then we assign roles for each person's permissions to each object. So, let me give you an example.

ABAC

ABAC stands for Attribute-Based Access Control. This is an access control model that's dynamic and context-aware, and uses if-then statements to decide on what permissions to use. So, the idea here is that we have something like, if Jason is in HR, then we'll give him access to the file server that contains the HR files. Attributes are going to use these tags and dynamic authentication to combine different attributes, and they can do this using all sorts of different software automation, and this is one of the newest forms of access control. It's not heavily used in a lot of places yet but it is trying to gain a lot of traction. The idea here is that we can look at Jason and we can start saying, is he part of this, is he part of this, is he part of this?

And if so, we can get a consolidated list of all the things Jason can do and give him permissions to that. That's the idea with attribute-based. When you think of attribute-based, I want you to think of dynamic authentication and tags because this is all about tagging things so you can give them the right permissions.

Best Practises

The first one is known as **implicit deny**. All access to a resource should be denied by default and only allowed when it's explicitly stated. So, for instance, when I create a new folder, by default, I want it to not allow people to have access, and then I would go ahead and give access to the people that need access to it. This gives you a higher security environment.

The next one we want to think about is **least privilege**. Users are only given the lowest level of access they need to perform their job functions. This kind of goes back to, do I want them to be a user, a power user, or an admin? Well, if they don't need all the functions of an admin, then don't make them an admin, make them a power user. If they don't need the power user, then make them a regular user and take away as much permissions as you can to give people the least they need.

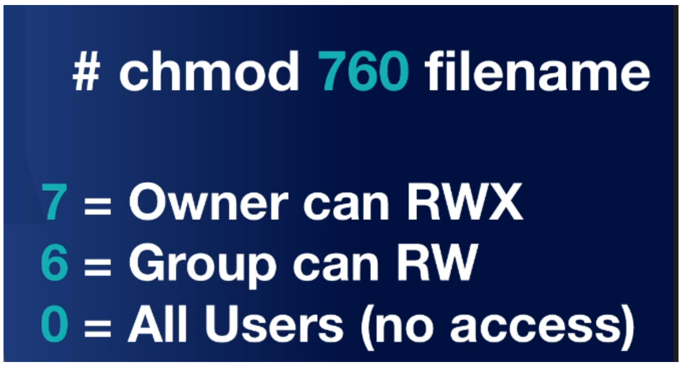
The next best practice is maintaining a **separation of duties**. This means that you're going to require more than one person to conduct a sensitive task or operation. So, I'll give you a great example of this from the corporate world. If you look in the corporate world, they have lots of money in their checking accounts, right? Well, who's going to be able to sign the checks for a corporation, and does it only require one person or two? Let's say that they were going to write a check for $10,000, that should probably have two signatures so that no one employee can write themselves a check for $10,000 and steal it.

Now, the last best practice I want to talk about here is called **job rotation**. This occurs where users are cycled through various jobs to learn the overall operations better, it reduces their boredom, it enhances their skill level, and most importantly, it actually increases our security. Now, how does all this happen? Well, job rotation helps the employee become more well-rounded and learn new skills.

Users/Groups

Now, permissions in Windows can be set into multiple ways. You can have full control, you can have modify, you can have read and execute, you can have listing of the folder contents, reading or writing.

Now, when we look at our permissions, they're usually broken down in Linux into three categories. Read, write, and execute. They're much simpler than what you saw there in Windows. Now, when we look at this, though,   we actually have these assigned to our owners, our groups, and all users. And for some reason, owners is called U, groups is called G, and all users is either symbolised by O or A.



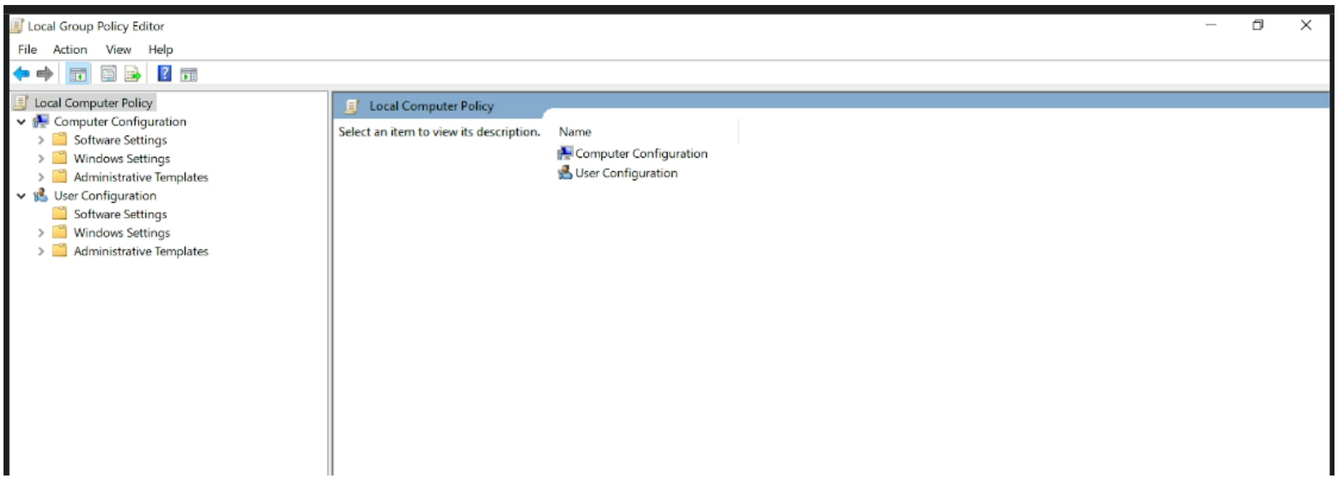
```
# chmod 760 filename
```

7 = Owner can RWX
6 = Group can RW
0 = All Users (no access)

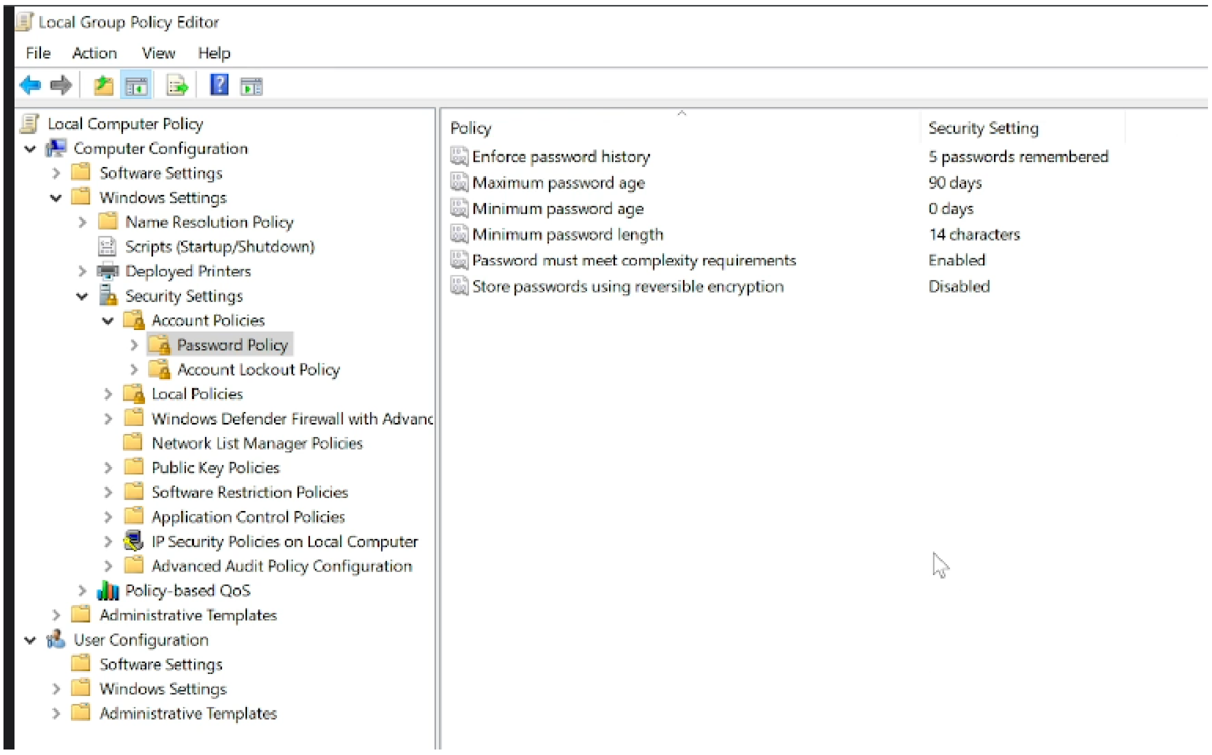
Permissions

Permission inheritance is going to happen by default. Whenever a new folder is created, it's going to inherit whatever the permissions are of the folder above it, which is called the parent. Now, the idea with inheritance is that whatever the parent says, the child is going to follow. And so, if the parent folder has permissions added or removed from it, guess what's going to happen? The child is going to have those added or removed from it, as well. This is a default action inside the Windows operating system. Now, when it moves from the parent to the child this is called propagation. Propagation occurs when permissions are passed to a subfolder from the parent through inheritance. There are two key things to remember though, in case it's been a while for you since you've taken the A+. The first thing is when you copy files. Let's pretend that I am copying a file or folder and I am copying it from C drive to a USB thumb drive. What will the permissions look like on that new copy? Well, when I move the file from the hard drive to the thumb drive, if I copy that folder, then permissions are going to be inherited from the folder that it gets copied into. Whatever its new parent is. Basically, it's going to lose its existing permissions. Now, if instead, I moved it from the C drive working folder to the C drive archive folder, they are both in the same hard drive and I move it instead of copying it. What do you think is going to happen this time? Well, if you move a folder, then the permissions are retained from its original permissions. And so, whatever that original parent was that it inherited from, it's going to take those permissions with it to the new folder. This is an important concept because if you forget the fact that when you copy something, you are getting new permissions, you can actually lighten the permissions of what it was and people can access documents that they weren't suppose to.

Policies Demo



Select an item to view its description.



Policy	Security Setting
Enforce password history	5 passwords remembered
Maximum password age	90 days
Minimum password age	0 days
Minimum password length	14 characters
Password must meet complexity requirements	Enabled
Store passwords using reversible encryption	Disabled