CS203 Gene's Class - Access Control Summary Request

LEC11-access-control.pdf
PDF

this is another slide from the course

Thanks! I've read the full slide deck: **LEC11 – Access Control** (dated 6/5/25). Let me know how you'd like to proceed. You can ask for:

- A **summary** of the full lecture
- An explanation of specific **slides** (e.g., "explain slide 11")
- A comparison between **models** (e.g., DAC vs MAC, ACLs vs Capabilities)
- A deeper dive into **RBAC**, **Bell-LaPadula**, or **Biba** models
- **Exam prep** questions based on the lecture

Let me know what you'd like next!

it's the transcribed lecture

I thought you might benefit from knowing a few things about an important area called access control.
which we touched upon, but only a little bit.
and then we'll go on our merriway.
So access control is one of those basic security services. Remember, in the very beginning of the class, we went in right, this overview of security.
And
we talked about all these services like confidentiality, like basically or secrecy. We talked about integrity, right? Authentication. Right? You know all these things already. Right and and availability, of course, important right? So this is what denial of service tries to do is essentially attack on availability. And then
there's access control and authorization. Right? Let's just call it access control. And what it's all about is.
how do we make sure that the right people get to, or the right
entities get to use the right resources, and only that.
in other words, how to prevent abuse or misuse of resources.
And so to do that, we need to control access to resources.
That's why it's called access control.
If you want to go into a little formalism. Access control is actually a very serious subject of study in in computer security. It's an active area of research.

By for the last 40 years at least.
It. It very much predates the Internet, because access control has nothing to do with by itself. Sort of it's not specific to distributed systems. Even access control is applicable to even a single computer that is used by multiple users. Right?
If you only have one computer
that is disconnected from everything, sitting on its own lonely computer. And you only have one user.
There's no access control, not new left.
Everything is yours right.
But the moment you have multiple entities, multiple users
having access to the same
physical resources, or, you know, not even physical, you need access control.
And so what it is is really a language, a language for expressing policy.
what can access, what, when, how, etc.
In order to enforce, to start enforcing access control. You need to 1st understand.
What is it that you're trying to protect.
like, understand the universe of things that must be protected?
You also need to identify those things or entities that might use those things.
No, they're called subject and object.
Object is a resource. Subject is an entity that wants to use potentially that resource.
And then this is the hard part need to specify rules
on the relationship between subjects and objects.
But so far so good.
And then, not only that, that's not enough. You need to enforce them now remember Carboros fondly.
Of course, Herberts. So, removing Herberts.
we had this central entity as Tgs, Tgt, Kdc. Right? It had the 3 names. It's authentication, server, key distribution center and ticket granting server right. And it all lived essentially 3 functions. But they all lived in the same box. Right?
And it was, or it is, among other things, doing access control.
So one type of access control is when the user logs in and the authentication fails.
The server says, no access for you.
That's an access control decision. You failed authentication.
Another possibility is that you didn't fail authentication. But the access control server says
you're logging in, but it's now 2 Am. You're not allowed to log in during the night.
That's a possible policy
that Joe and or I know Alice is allowed to log in between 9 Am. And 5 Pm. Otherwise no access for Alice.
Bob might be doing a night shift. He might be a night shift operator, and he's only allowed to access the system between midnight and 8 Am.
That's a policy, and Bob may be denied access, even if he supplies the right username and password in terms.
Let's go further.
Alice logged in, got a ticket granting ticket from Gerbert right successfully logged in.
Now Alice wants to access a service, remember services. Right?
So let's say it wants to access a.
Let's think about something. Fancy a 3D. Printer.
Okay, that says actual physical device. But there is a software running on it, right?
So she she goes to the with her Tgt to the Kerbers, Tgt tgs, and says, Hi, I am Alice.
Here's the Tgt that I obtained previously. I would like to use the 3D printer
while Tgt. Again performs an access control function and says, Is Alice allowed to use this service?

Maybe it's an expensive service. Alice is some mailroom clerk. What the hell is she doing? Trying to access a 3D. Expensive? 3D. Printer not allowed, or it could be. Alice
is not allowed to access this at this time, so no ticket for you.
Bob may choose to access a a scanner, something. Okay.
He goes to the Dgt. Presents his Tgs. Presents his tg, and says, I'm Bob, I want to access the scanner. Sure, no problem. Here is your service ticket. Remember the interactions. That's an access control decision that Dgt Tgs performs.
So you've already inadvertently kind of see access control in action.
Now, there is a bit of a obscure terminology around here. So the what are subjects and options.
Well, let's start 1st with humans, right?
Most of the time. We have humans that use computers in one way or another. The humans have accounts.
but notice that human usually is one right, like one individual human.
But an individual human can have more than one account. Man
and one account is generally not owned by multiple humans.
If it is, it's a bad idea, really bad idea.
Okay, I don't know of any justification for more than one human to own the same account.
So sorry.
Yeah, you still use the same password.
Right? So in some ways it's like it is one human, but it has different profiles.
Yeah, it's true. But in a, in a real system, right? This is an application streaming service, you know. All they care there about this billing. Right?
You're not doing a resources are like, I mean. Another example might be more Amazon prime.
where you know with Amazon Prime. Anybody use Amazon prime
when you log in and it will tell you.
oh, these are movies are free, and these are not right. And there's like this tier like, anyway, let's get back to the real systems. So that is like I said, human can own more than one account. And that's why we refer to accounts as principle, right? This web
that account is called the principal.
And then subjects in access control are essentially programs, processes running on behalf of that principle. So whatever computer, you multi user computer, you're running right?
There are sets of processes that are currently executed or currently supposed to be running right active process.
And if you look at your windows, Mac and Android, you will see if you know how to list the running processes, you will see that every process has an owner.
and that owner corresponds to what I call principle, the user account.
Okay? So these programs or processes, we're gonna call them subject. They are the subject and objects are resources.
Okay, typically, like hardware resources, printers, scanners. 3D printers
I don't know. All kinds of attachments and peripherals. Network interface is a resource.
A keyboard is a resource. A mouse is a resource.
Okay, they're all objects, cameras attached to your computer or separate routers or resources. Okay, disks or resources.
Okay, their object.
So access control. It really is focused on what a subject is allowed to do.
And conversely, what may be done with an object
very trivially.
There are
kind of read or look at type operations, and then there are change type operations or write type operations. And some of this may be surprising to you
like, read a file that makes sense right? We all know. Read, the file is like looking at something.

You're not disturbing it in a way. Right? You're just looking at something.
Read them checking a printer. Queue might be a bit surprising.
So you have a printer that is shared right
who is allowed to check the printer status in the queue of jobs viewed on the printer.
That's essentially a read operation because you you are reading the status of the printer.
And typically, actually, it involves looking at some file or on some driver status, on, on that, on the computer that runs that printer to see what the queue is like. So essentially, we
read the screen reading a screen of the device, pulling things from the screen.
That's also a redeberration.
Query, a database!
Turn on the microphone.
What changes an object is like this? Right? That's clear. Right into a file, overwriting, inserting, appending, deleting a file. Those are all right operations.
They're changing the option.
Printing on a printer changes the object it does.
You are writing to a printer when you print.
Surely right.
You're also consuming resources called paper
displaying on the screen. You are writing to the screen
right? In fact, if you've ever written like unix style programs. So you know, when you write to a screen directly to the screen, you're writing, you're connecting to the device driver
with the screen. So you're actually sending data to be displayed. That's writing.
Using. A speaker might surprise you. But using a speaker is writing because you are sending audio through the speaker
you are writing to it just like an audio is a kind of a printing right? Just without paper.
Sending packets, is writing.
You're writing to the network
doesn't matter if it's what kind of network this is. Wi-fi wired Ethernet. 5 direct fiber. Whatever you are writing to the network, right? You're going through the device driver to the network. You're writing packets. That's a right operation.
Typically, you'll find more than that right? I. The previous slide is a bit simplistic, right? Looking at something and changing something. But there's more right in many systems. There's a distinction. There's, of course, read and write, but some systems differentiate between writing in general and appending.
So, for example, if you have a log
like database log, or just a system, activity, log
writing to a log is one thing
appending to logist is very different, and in many cases
writing is not allowed, but the pending is
because writing is a superset of appending
by by allowing somebody append only access. You're saying you cannot change what is already in the log. You can only add to it.
and that's very important for security, not outside of access control. Just to know, for example, what happened in the system right? A lot of times to detect
intrusions, malware infestations, ransomware infestations. It is critical to look at the log.
Okay?
And a lot of times, if you allow writing to a log, you are basically you're screwing yourself completely because writing to a log is deadly. It means you can overwrite anything.
But a pending is different, right? You cannot change history, you can just add to it.
and then finally execute right execute is an important right. You might be able to write something, you might be able to read something, but executing it is different, right, because execution

fundamentally, is a very different different
privileged them. Writing and and reading
execution has other side effects right
like, for example, allowing you to download something from the web, like many operations, will do this. I say, Mac OS, that you can download whatever crap you want from the web.
you can read it you can.
You'll eat it the right way.
but if you try to execute it, it will pop up and say, Hey, you brought this from the. This is dragged in from some Godforsaken place on interest. You sure you want to execute it?
Have you seen that message?
Many operating systems will do that? I'll just say, I'm sorry I'm not executing this right? Not executing.
because I don't know what the heck this is
on unix. You basically have 3
right? Yeah, yeah. Execute. Really. Now, on no, on vanilla unix. There are some extended unises out there, flavors of unix that allow you more more than that. But generally you'll you'll see, like XR. And W.
On fox right when you do.
Directory listing on files, you just list the file. You will see that it has these 3 flags right
in several incarnations.
but it's essentially execute, read, write, you might have execute access. No, read, write, you might have read, but not execute and write, and all the combinations there off. Some of them don't make sense, but you will see them nonetheless.
Okay, I saw unix access control requires its own lecture. If you don't know how it works. There are plenty of very easy tutorials online. If you wanted to understand how access control works on. So there are basically 2 types of access control out there. Any of you ever worked for a government organization?
Nope, okay, you did where the court. Yeah.
Federal state state. So if you use their system.
did, you have much freedom about making accessing things?
You were basically put in a straitjacket, weren't you? Yeah, right? So this is how government works. A lot of corporations work. With this, they employ what's called mandatory access control, where some system administrator or some authority says, access control will be like this.
Okay?
And you, as a user, have very little choice about what you can. What you can do. You are given certain access rights, and that's it.
Now, in a way, we have that here at Uci.
I say, in a way, because it's you might not see the analogy, but you know how
we are mandated to use. VPN.
We have been recently mandated to install spyware.
Excuse me, Trellis right.
Have you ever heard of it?
No, you will hear about it.
Well, many of us refuse to do it, but
essentially for us the faculty and staff, in order to access certain services like docusign and payment systems. And our accounting system, we have to not just use VPN and go through multi-factor authentication. We have to also install this what's called endpoint Edr tool called trellis, with essentially spyware.
That's heavy handed mandatory access control, which of course, most of us hate, and I gotta fight against.
But I'm kind of surprised. I guess most of you are not subject to it unless you work at the university,

and I don't mean like a Gsr, I mean, like you have some other like a part-time job
anyway. So the idea is, the mandatory access control is imposed on you as a user
and in a distributed systems where all users generally have a decide on their own. Like, for example, you have your own laptop. Maybe you share it with somebody. Maybe you don't. But basically you set your own access rights right? You decide what you protect.
And that's called discretionary. Because
user has discretion or the sorry, the resource owner. Right? You own the computer, you can dictate your own policy.
There are 3 types of kind of a data structures, if you will, that you can use for access control.
They're very simple access control matrix.
Sorry that should be Acl.
okay, never mind it, says Mac. In both places. But it should say, acm, access control, NASA, matrix Acl or access control list and capabilities. Right? So the capabilities are best for discretionary access control and
matrix and lists are best for mandatory access control.
And I'll give you some detail in a minute.
So an access control matrix is basically what you would imagine it to be. Imagine a matrix that lists as columns, all resources that you ever want to protect, and rows are the subjects.
Okay? So I'm simplifying it. Remember, I said, subjects are actually processes running on behalf of users. But I'm simplifying it here. So when I say Alice and Bob, I mean all processes running on behalf of Alice and all Bob, all processes running on behalf of Bob.
So we have in this case a very trivial, very unrealistic matrix. Because it's tiny. We only have 3 columns and 2 rows.
We have 3 documents we want to protect. And they're objects. And Alice has no rights to access. Build. Doc
just has no end.
0 empty set.
There is a program called Edit dot exe, and Alice has the execute right to that program, but not nothing else. As far as fund.com she could execute or read.
Not a problem.
Bob, on the other hand, can read and write his build. Dot, Doc.
Notice you cannot execute it. Why? Because build that, Doc is probably not an executable file.
He can execute edit Dot, Dfc.
no point in reading it right? It's it's machine code, right binary
or writing to it because it shouldn't be able to delete it
or change it, and as far as fund.com. Well, probably Bob has authority over that, because he can execute regularly and right. So maybe it's a program that Bob wrote, and he's just making it available. Well.
now, this is clearly not realistic, right? The tiny matrix in the real world and matrix can be humongous.
So it is used in the real world, but in certain very specific circumstances, usually in a government type, military setting. But it does not scale. Well.
because in the real world you might have thousands of millions of objects, and maybe.
let's say, millions of objects and thousands of subjects.
and guess what this matrix will look like.
You will have mostly empty spot slots.
Oh, things like empty set there with Alice and Bill Dot, Dot, meaning no access.
So what that means is, the matrix will be very sparse and ginormous.
So it's not a very good way of storing just consumes a lot of space and traversing it is hard.
It's like a giant database.
An alternative is access control lists, access control list.

Basically, you tell each object who can use it.
So each object essentially gets a list
of subjects that can use it and how they can use it.
So remember the matrix from the previous slide, the same matrix.
If we convert it to access control list, we'll say that. Okay, there's a build dot. Doc File stored on this and right next to right inside the build out, Doc. There is an access control list that says
Bob can read and write.
It says nothing about Alice, because Alice has no accent.
So the only thing that Bill Dot Doc knows is when somebody tries to open it and says, Are you, are you, Bob? If you're not Bob piss off.
If you're Bob, are you reading me? Are you writing me? What are you opening me for? Reading or writing?
Okay, execute? No, whatever is not allowed explicitly is not is is forbidden
for edit dot efc. That file will store the fact that Alice can execute and Bob can execute.
Charlie wants to do anything to that file. Forget it.
No reading, no writing, not even for Bob and Alice, just basic
and for fun.com. Well, Alice can execute and read, Bill can execute, read and write everything else forbidden.
Now, what do I say? What do I mean by oh, there's a list inside the file. It's not really inside a file. It has to be associated with that file
stored somewhere alongside it.
As you can imagine.
there'll be as many acl as many lists as their object right, because every object has to have its own acl.
and, most importantly, each acl must be either signed
and attached to that somehow associated securely with that object, or it must be stored in some very protected space.
And who is going to enforce it
right? Who's gonna enforce it? When, when let's say, you got command and operating system command goes open
build dot dog to execute right?
Who's gonna enforce that right? Who's gonna check whether it's allowed. Well, that's usually done by the operating system.
There's a component on the operating system called Reference monitor.
Anybody took a serious OS class anywhere.
Hmm, oh, Jesus, really.
Okay, anybody took a Mickey Mouse OS class anywhere.
Oh, more. Okay. Must be taken here. No? Okay. Well, in a real OS class, you would have learned about reference monitor, because it's like an important part of the of the operating system. Most operating systems have a reference monitor, including even android
and it essentially, it performs this access control function. So it's not enough to like specify. Oh, this file can be. Tell each file, or you can be opened by this, you can be executed by that, etc. No, somebody has to enforce it.
and that's the operating system's job
or the reference monitors. Component job is to enforce it in real time.
So the Acl. Is A is a reasonable approach.
generally preferred approach when the number of subjects is a bit quite a bit less than the number of objects.
Why? Because then, Acl, associated with this object doesn't grow huge.
right? Because the number of subject is relatively small.
Also, you can. If you imagine that you have a file system right? Because everybody understands. File

this here. I have examples with files, but it can be other types of resources. But imagine you have a directory structure like you have a most, all modern systems. Right? You have directory structure.
You can assign an Acl to a directory.
So you don't actually have to like assign individual acls to individual files.
You can just say this directory right? Has an Acl.
and it applies to all the files within that directory. Does that make sense to kind of a reduce the amount of work right? You need to do and reduce the number of Acl.
So if 100 files in the Directory, ABC, all have the same Acl, there's no reason to store an Acl. For every 100 over the 100 files. You just store it in the Directory itself.
because, as you should know, and I hope you all know what is a directory.
What is a directory?
Hello, in a file system? What is a directory?
It's a file.
It's itself a file.
Yeah.
When you do a Dir in windows or Ls in Unix.
What are you doing?
What are you actually doing?
You are reading the Directory.
You are opening the there's a directory file. When you're doing Ls or Dir, you're opening the directory file and reading it
right? So it dives on the screen.
When you create a new file in the Directory. What are you doing then?
What you are writing to a directory file
directory is just a file. It's a very special file, but you're you can read, you read and write from. You cannot execute a directory
most of the time at least. I've never heard of anybody executing a directory. Okay.
so what I'm saying is that acls are not as bad as they as they sound. If you have a whole bunch of files that all have the same acl, right? Identical.
Then you just don't bother assigning an Acl to these files you are signing to the Directory where those files live.
That's it.
And it applies to everything inside the record capabilities.
Capabilities are a 3rd option, and it's the inverse of or the opposite of, Acls
here, with capabilities. Objects don't know anything.
They don't store any rights, but subjects carry a capability
or a set of capabilities that tell those subjects what they can or cannot do, with what objects, so the burden is placed on the subjects.
not on object.
So
again, remember the access control matrix. If we wanted to express the same policies as the matrix has pressed in capability format.
Alice's capabilities will be 2. She has 2 capabilities. Edit dot A/C she can execute fun.com execute. Read. Bob has 3 capabilities.
You can clearly see that this is equivalent.
So it becomes Alice's and Bob's burden to keep those they lose those capabilities, no access.
Okay, anything that is not in the capability is not allowed.
Now, how many are there? Well, there are as many capabilities as there are subject object there huge number potentially
again, just like with Acl, each capability
must be signed or otherwise securely protects are protected.

Because if I give him a capability well, he can just change it right? They say. Well, you can access my office between hours of 9 Am. And 5 Pm. He can change it to midnight to 5 Pm.
Capability must be protected.
You are all familiar with capabilities. You have used capabilities in the physical world. Probably as long as you've been living.
A driver's license is a capability.
a passport is a capability regardless which country issue okay.
in a way, a birth certificate is a capability.
They are the driver's license and passports are really good examples. Also keys, physical keys, our capability
building access cards or conference room access cards that some of you have our capability, you carrying with you.
Right.
You carry with it there your capability. They allow you to open doors. The door does not ask who you are.
It just says you have the key. I will open for you.
The key is the capability supposed to be secure.
If you have a good keying system, it will be secure.
Okay, I have a card like some of you, and that card is programmed
to allow me access to brand whole after hours.
It also allows me access to conference rooms in Brent Hall that are equipped with the card readers.
Some of you may have, though. Do any of you have those cards.
If you have them, they're probably not gonna allow you access to conference rooms, but maybe to Brenholm.
In your case, some engineering building. Right? Okay? Same idea.
We we may have very similar looking cars, but they allow us different things.
Those are capabilities.
They are stored with object. With subject. I am a subject.
I carry the capability, but it is secure. Because I yeah, the the presumption is, I can't hack into this card
and modify my capability.
So the the problem with capabilities they're hard to revoke.
Yeah, capability like a badge. Right? The employer badge
the one that access allows you to pass by security desk, or the one that you have to scan.
maybe hard to revoke. You need to notify everybody that this capability, no longer valid
requires the owners of objects to keep track of capabilities.
That's why we have typically an It department with security people right there, or at least some security specialists in the it department that manage this
generally, capabilities are preferred when the number of subjects is greater than the number of objects.
it's easier to manage them like
what would be an example. An example is, think about how many conference rooms there are in like a brand hall.
My last time I checked there were 2 on each floor, so that 2 to 1011, 12,
something. Blast the front doors.
I don't know. 13. Let's say there are 20.
The card equipped doors in Brent Hall. How many people use Brent Hall?
Well, over a thousand, I would say these hundreds.
Do you see? What do you see? Any? The analogy here
we, the people who use our subjects.
the conference rooms are objects. The number of objects is smaller than the number of subjects.

Capabilities are pretty good here I have.
because every subject has the capability, and there are only a few objects that they could potentially have access to.
So this is kind of a over generalization. But typically in centralized systems.
And I, when I say centralized, I don't mean like there's a giant mainframe or something like that. No, I mean the centralized systems that are like a centralized organizations, right? Like government, like, you know, Federal courts, state courts, police departments any kind of government agencies or large corporations
use mandatory access control, and they often use acls, right access control list
or matrices, and then more distributed systems that are kind of a more liberal and free for all Discretionary access controls and capabilities are better, and we live in it. We here in Uci live in a kind of a blend we don't strictly fall into like a rigid government or corporate structure, and we don't actually fall into this completely freewheeling world of anything goes. So we kind of have a combination.
Last thing I want to talk about is just a slight twist on access control called role-based access control and role-based access to a very trivial idea.
Main idea is here is that instead of managing users individually, and this actually applies to more users than, let's say, sub than processes, or what I defined earlier as a principle, but more like users. Users are associated with roles. For example, your role is student.
My role faculty staff is another role. Administration staff
contractors is another role. Right? People who I don't know clean, clean the offices, right? They're not employees. They're generally contractors.
These are different roles we have on this campus. I'm probably missing a few, but there's not so many
now if, instead that the main idea of role based access control. If, instead of looking at each individual and saying, you can access this this and that.
there, there and there, we say, you know, we don't care about that individual. What we care is your role.
And instead of assigning rights to an individual, we assign rights to the role.
All students can do this, and only this. All faculty can do this and that, but not more. All Staff can do something else. But not okay. Get the idea.
That's it. That's the whole notion. Right? So instead of treating individuals, I know it sounds very dehumanizing. But in access control, that's okay.
We're not. We're not politicking here. It works well, because in most organizational settings we have roles right?
Even on distributed systems, you know, like, anybody use reddit.
Yeah, great, are you? Occasionally. Any of you are redditors.
You know what redditors are right.
There's special people in charge of a topic, right? You have to earn the right to be a redditor. That's a rule. Otherwise, you're just a casual user. You can use Reddit without an account. You're a casual user. Once you create an account, your your role is elevated. Same with Wikipedia, by the way, also distributed platform. But it has roles
right? There are editors, Wikipedia. Then they're casual users, right?
The users with accounts without accounts. And they're different roles. So main thing is to remember is a user has permission only if if it has a role. If a user has a role associated with that permission.
So here's a, here's a i'm probably about by example. So suppose we're talking about cops.
Okay? So in this case Bob Dean and Charlie have cost.
and because they're tops this one policemen, they have access
to the station police station. They can enter the police station. They can go to the locker and get the uniform. They're allowed to access the uniform, and they have access to weapons.
and every one of the 3 of those guys has separate access station uniform weapons.

That's how access control works. Right? Doesn't matter if this is done as an acl acm or capability.
Now, why, if, instead of that, we created a role, cool, cool policeman.
And then we said that anybody who has that role associated with them
right away has access to station, uniform and weapons.
Now, what where else would you have in a police station?
Well, you have admins right in the police station secretaries
blood spatter specialist like Dexter, you know. That is.
I know all kinds of scientific types, right? That people, even sometimes people with Phds who work there analyzing, you know, forensic and things right? They're not cops.
They don't have badges, but they work in a police station. They have different roles
and different. They can access. Weapons, for example, can't wear a police uniform.
but they have access to the station, so cops are only one of the roles that is present.
So that way you see how the information becomes compressed.
instead of having many arrows like you saw before between Bob, Charlie and Dean, each arrow going to station uniforms.
Only now we have these arrows relationship. Bob is a cop.
Charlie's account. Dean is a cop, therefore every one of them has access to all 3 resources
not complicated, right?
Maybe sleep inducing, but not complicated.
But here it doesn't work. So, for example, if you have a cop
who has to, because he's on probation.
and that cop can only access station, but cannot wear a uniform because he committed some offense. But he's allowed to go to the station. I don't know and read a book and sit there, but not wear a uniform or carry weapons. Then this doesn't work right, because then you have different types of cops. Then you need to create, maybe a separate role called
suspended cops.
You follow.
So it only the art back works. When you have well defined, roles that where any member of that role anybody who has that role has the same rights.
So if you have a new cop right, Alice joins the police force that it's easy, right? It's easy to make it make her cop. You just give her a capability, right? Or something like our badge that says, Hey! With this you are allowed to access the station. With this you're allowed to access uniform and weapons.
That's it.
Oof enough.
I had enough. You hit enough.
Have a good summer.
Well, after Tuesday, boys 8 am. Sharp.
8 am.
That's what it said, didn't it?
Oh, I just thought I had to check the way. If you're awake so
good. Good! 1030 to 1230, I think so. You have 2 h, remember, and the same rules as in the midterm. No devices, or I much prefer you write in 10,
then pencil. I much prefer you write allegedly, because, as I already said to some of you.
I will take off points for encrypted writing. I had to really work hard to decipher some of it last time.
All right. See you Tuesday.

Got it — I've now integrated both the **slides** and the full **transcript of Lecture 11 on Access Control**. From now on, any question you ask about this topic will be answered using both sources.

You can ask things like:

- “Explain DAC vs MAC with examples.”
- “What’s the intuition behind capabilities?”
- “What problem does RBAC solve?”
- “Why is access control matrix not scalable?”

Just let me know what you want to dive into!

always answer from both slides and especially from prof's lecture

Understood — all answers about access control will now draw from both the **slides** and especially the **professor’s lecture transcript**. Ready when you are!

firstly, summarizee the prof's lecture

Here’s a detailed **summary of the professor’s Lecture 11 (Access Control)**, based on the **transcript** you provided:

🔐 Access Control: The Big Picture

Access control is one of the foundational security services, alongside:

- **Confidentiality** (data secrecy)
- **Integrity** (prevent unauthorized alteration)
- **Authentication** (verify identities)
- **Availability** (ensure resource access)
- **Access Control** (ensure only the *right entities* can use the *right resources*, *in the right way*).

It's about **preventing misuse** or **abuse** of resources by controlling access. This is relevant even for standalone systems with multiple users and long predates the Internet.

🧱 Access Control as Policy Language

Access control provides a **"language" to express policies**:

> Who can access what, when, how, and under what conditions?

To **enforce access control**, you must:

1. Identify resources (objects) you want to protect
2. Identify entities (subjects) that may access them

3. Define subject-object interaction rules (access policies)
4. Enforce these rules **at runtime**

👩‍💻 Terminology

- **User**: The human being
- **Principal**: The user account (a user may have many)
- **Subject**: A process acting on behalf of a principal
- **Object**: A resource (e.g., file, printer, camera, scanner, etc.)

Example:

> A subject could be a process launched by Alice's account to read a file (object).

🎯 What Does Access Control Regulate?

- **What a subject is allowed to do**
- **What may be done with an object**

🛠️ Access Modes

Grouped into:

- **Observe** (Read): e.g., read file, screen, printer queue
- **Alter** (Write/Append): e.g., write to file, print, display to screen
- **Execute**: Run programs or commands

Important notes:

- **Appending** is *not* the same as writing. Appending allows logging without modifying past entries — crucial for forensic analysis.
- Execution often requires stricter controls due to potential side effects (e.g., downloading malware).

🐧 UNIX-style Access Modes

- **Execute (x)**: Run a file or traverse a directory
- **Read (r)**: View contents or list directory
- **Write (w)**: Modify, append, or create in directory

These show up in `ls -l` as flags like `rwx`.

🧱 Types of Access Control

1. Mandatory Access Control (MAC)

- System-wide policy dictated by administrator
- Example: Government agency, large corporations
- Users **cannot** change permissions

  UCI's VPN, MFA, and spyware-like endpoint monitoring tools (e.g., Trellis) are modern examples of MAC.

2. Discretionary Access Control (DAC)

- Resource **owners** (users) set policies
- Common in distributed, personal systems (e.g., your laptop)

📊 Data Structures for Access Control

A. Access Control Matrix (ACM)

- Rows: subjects (e.g., Alice, Bob)
- Columns: objects (e.g., files, printers)
- Cells: access rights (e.g., read, write)

✅ Conceptually simple, ❌ impractical at scale (very sparse and large)

B. Access Control Lists (ACLs)

- Stored **with the object**
- Lists which subjects can do what
- Easier to manage when **#subjects ≪ #objects**
- Common in file systems. Directories can share ACLs with all contained files

C. Capabilities

- Stored **with the subject**
- A capability = permission to access an object in a certain way
- Real-world analogies:
  - Keycards, passwords, passports
- ✅ Scales well when **#subjects ≫ #objects**
- ❌ Harder to revoke in distributed systems

💡 Who Enforces Access Control?

- The **Operating System** via a **Reference Monitor** component
- It checks policies in real-time before allowing access

👮 Role-Based Access Control (RBAC)

- **Users are grouped into roles**
- **Roles are assigned permissions**
- Simplifies policy management: assign rights to roles, not individuals
- Example:
  - Role: Cop → Access to weapons, station, uniform
  - Users: Bob, Charlie, Dean all inherit cop permissions

RBAC is efficient when:

- Roles are well-defined
- All users in a role share the same permissions

But **RBAC fails** if fine-grained control is needed (e.g., one cop can't access weapons). In such cases, sub-roles (e.g., "suspended cop") are needed.

📃 Final Notes

- OS-level enforcement is essential (especially Reference Monitors)
- Use **ACLs** when fewer subjects
- Use **Capabilities** when fewer objects
- RBAC is ideal for **organizational systems** with clear roles

---

Let me know if you'd like this split into a printable handout, diagrammed version, or quiz format!

> explain slide 15

📄 Explanation of Slide 15: "In Summary"

Slide Title: **In Summary**
Content:

- Centralized Systems:
  – MAC and ACLs are better
- Distributed Systems:
  – DAC and Capabilities are better

🧠 Professor's Lecture Explanation:

The professor summarized this distinction clearly and emphasized **why different access control mechanisms are preferred based on the system's nature**:

🔵 Centralized Systems

- Examples: Government agencies, large corporations, university servers
- Characteristics:
  - Central authority (e.g., sysadmin) manages all access
  - Strict, uniform enforcement of policy
  - Users have **limited discretion**
- **Preferred mechanisms:**
  - **Mandatory Access Control (MAC)**: Policies imposed top-down, users can't override them
  - **Access Control Lists (ACLs)**: Each object has a list specifying who can access it and how
- ✅ Why it works:
  - Fewer users (subjects) compared to objects (e.g., documents, services)
  - Easier to secure and audit
  - Stronger control and compliance

  🗣️ *Example from lecture*: UCI's mandatory VPN, Trellis (EDR software), and MFA to access payroll/documents — all examples of MAC. The user has no control over these policies.

🟢 Distributed Systems

- Examples: Personal devices, collaborative platforms, decentralized systems
- Characteristics:
  - Individual users (resource owners) manage access to their own resources
  - More flexible but riskier
- **Preferred mechanisms:**
  - **Discretionary Access Control (DAC)**: Owner sets the policy for their resources
  - **Capabilities**: Each subject carries tokens (capabilities) granting access to objects
- ✅ Why it works:
  - Usually more subjects than objects (e.g., many users accessing a few shared files or rooms)
  - Capabilities scale better since each subject keeps their own rights
  - Allows flexibility and user autonomy

  🗣️ *Example from lecture*: Physical access badges for rooms in Bren Hall — many users, few objects. Capabilities (keycards) are easier to manage than object-bound ACLs.

✅ Summary

System Type	Preferred Mechanism	Why
Centralized	MAC + ACLs	Top-down control, fewer users, stronger enforcement
Distributed	DAC + Capabilities	User autonomy, many users, easier to manage at the subject level

Let me know if you want a diagram to visualize this comparison!

explain slide 18-22

Here's a detailed explanation of **Slides 18–22** of Lecture 11 on **Access Control**, with the professor's live commentary integrated.

🔹 Slide 18: Example – Cops (User/Permission Association)

Shows a table with 3 users (Bob, Charlie, Dean) and 3 resources (station, weapons, uniform), with individual access links drawn.

🧠 **Lecture Insight**:

- The professor sets this up to illustrate **basic user-permission mapping** (before RBAC).
- Each user is explicitly assigned permissions to specific objects.
- **Downside**: This grows quickly as users and objects increase.

✍️ *"Bob, Dean, and Charlie are cops... and each is linked directly to each resource (station, weapons, uniform)."*

🔹 Slide 19–20: Example – RBAC (Role-Based Access Control)

Diagram now introduces a *role*: "Cop."
Bob, Charlie, and Dean are connected to the *Cop* role, which is in turn connected to the 3 objects.

🧠 **Lecture Insight**:

- **Roles simplify access control**: assign permissions to the *role*, not each individual.
- In this case, all cops get access to station, weapons, and uniform by virtue of their role.
- This **compresses the number of relationships** needed to define permissions.

✍️ *"Instead of having multiple arrows from each user to each resource, you just assign the role. It's more scalable and efficient."*

🔹 Slide 21: When RBAC Doesn't Work

Same "Cop" setup, but implies a new constraint — **a cop who cannot access all resources** (e.g., can't wear the uniform).

🧠 **Lecture Insight**:

- RBAC **fails when you need exceptions** or fine-grained control within a role.
- For example: if a probationary cop **can access the station** but **not wear a uniform or access weapons**, you can't express this with just the "Cop" role.
- **Solution**: Define new roles like “Suspended Cop” or “Probationary Cop.”

  ✍️ *“It only works when every member of a role has the exact same permissions. If not, RBAC falls apart unless you create sub-roles.”*

🔹 Slide 22: Example – Alice Becomes a Cop

Alice is added to the “Cop” role, and gains the same 3 permissions (station, weapons, uniform).

🧠 **Lecture Insight**:

- **Ease of onboarding**: Just assign the role — Alice inherits all the permissions.
- This is **RBAC's core benefit**: simplifies administrative overhead when adding new users.

  ✍️ *“You don’t have to manually connect Alice to each resource. You just give her the Cop role — she gets it all.”*

✅ Summary of Slides 18–22

Slide	Focus	Key Idea
18	User ↔ Permission mapping	Each user assigned permissions individually
19–20	Role-Based Access Control	Users assigned roles → roles carry permissions
21	Limitation of RBAC	RBAC breaks when users in the same role need different permissions
22	Adding a user (Alice) to role	User inherits all permissions of the assigned role easily

Let me know if you want a diagram comparing **pre-RBAC vs RBAC vs broken-RBAC** cases visually!