Who gets access to what data.

Note: the configuration of security and sharing models works at the API level, meaning it is in effect regardless if the user is accessing via UI or API calls.

Levels of Access - Can control which users have access in your entire org, single object, field or even record.

Access to object level data is simplest to control. It does require grouping your users, though.

Limiting access to a field can be done even if access to its parent object is available to a specific user

Records- think comments/reviews.  You can see all, but only edit your own.
Four ways to manage access to records:

1. Org wide defaults - default level of access. This should be set to the most restrictive level, then use other tools to give access to select users
2. Role hierarchies - users up the hierarchy have access to all records owned by users below them. This does not necessarily have to match your org chart exactly. Think about roles defining levels of required access.
3. Sharing rules - automatic exceptions to org wide defaults for particular groups. They cannot be stricter than your org wide defaults
4. Manual sharing - allow owners of particular records to share them with other users.  This is a non-automated managing tool, but can be useful in some situations.

TIP: Make a chart. List types of users in org.  Ask What objects can they see, fields on object? What records should be hidden by default? What exceptions should be made and when?

Audit System:
Use auditing to diagnose security issues. Helpful tools include:

1. Record Mod fields - name of owner, creator etc. basic auditing info
2. Login History - successful and failed login attempts list goes back 6mos.
3. Field History Tracking - you can turn on auditing- auto track changes in values of fields. *all custom, but only some standard objects allow this.
4. Setup Audit trail - this logs when modifications are made to org configs.

Managing Object Permissions:

Simplest way to control data access is set permissions on types of objects. Control what groups of users can do to those objects.

Set permissions with profiles or permissions sets. A User can have ONE profile, and many permission sets.

Profiles determine accessible objects and things they can do with records for a user.
Permission sets grant additional access settings to a user.

Much like Org-wide defaults, set the profile access to the minimums for all users of that profile. Then, expand with permission sets

*Profile setting determine which data user can see, Permissions determine what the user can do with the data (create, edit, etc)

Profile functionality in your org is dependant on your user license type.

Manage Profiles:
Overview page can be found in ‘Setup’, search ‘Profiles’, select a profile.
Manage Permission Sets:
Overview page can be found in ‘Setup’, search ‘Permission Sets’, select a PS.

Create Profiles:
Easiest way is to clone an existing profile and then modify it to meet new needs.

- ‘Setup’, ‘User Management Settings’, then enable ‘Enhanced Profile User Interface’, click Save.
- Custom profiles are completely editable, but standard profiles must accept the permissions settings as they come.

Assign Profile:

- ‘Setup’, ‘Profiles’, select one to edit, click Object settings, click Edit.
- Set the most restrictive settings for the user type and save.
- ‘Setup’, ‘Users’, click Edit on the one you’d like to change
- Select the profile you just set up from the ‘Profile’ dropdown menu. Save.

Set Permissions: Two general purposes to do this.

- Grant access to custom objects or apps
- Grant permissions to specific fields

*Note: if permission is given via a profile, you cannot take it away with a permission set...you can only add more. You must remove it from the profile. If this affects other users you will need to include a permission set for those who still need the permission. Only goes one direction, in other words.

Managing Permission Sets:
Overview page can be found in ‘Setup’, search ‘Permission Sets’, select a PS.
Each Permission set is organized into:

| | |
|---|---|
| ● app settings | system settings |
| ● object permissions | field permissions |

Create a Permission Set:

1. ‘Permission Sets’ in ‘Setup’, click Clone next to the set you want to copy. *This will have the same user license as the original cloned PS. To create a set with a different license, click New.
2. Enter a label and a description. API name is unique, used by API and managed packages. Auto replicates the label but can be modified.
3. If a new permission set, select a user license option.
    a. If planning to assign to multiple users with different licenses, select --none--
    b. If all users getting this PS have one type of license, select that same type here
4. Click Save and be returned to the PS overview page.
5. PS toolbar, click ‘Manage Assignments’, click ‘Add Assignments’
6. Select the users to assign to this permission set and click ‘Assign’ - Review any messages on the Assignment summary page. Click Done.

Control Record Access:
Similar to Field Access, this is determining which records a user can see.
Questions to ask:

1. Should user(s) have access to every record, or just a subset?
2. If subset, what rules should determine whether the user can access?

*Record permissions are evaluated by object, field, and record level permissions. When there is a conflict, the most restrictive settings always win. (Ex: give edit access to an object, but read-only to a record- record will be read only. Give read-only access to an object, but edit to a record- record will be read only.)

Ways to control record-level access: (four)
**See page 1** - Org-wide defaults, Role hierarchies, Sharing rules, Manual sharing
Laid out in a reverse pyramid (org-wide being bottom/point) they are meant to open up access as you move toward Manual sharing.

Access is determined by the above and is based off the following key principles:

- User’s baseline permissions are determined by their profile.
- If the user has permission sets assigned, those also set the baseline permissions along with the profile
- Access to records not owned by user are set first by the org-wide defaults
- If org-wide defaults are anything **less than** Public read/write, you can re-open access to certain roles via role hierarchy
- You can use sharing rules to expand access to add’l groups of users
- Each record owner can manually share with other users by using the share button on the record

Questions to ask to determine Org-Wide defaults you need for your App:

1. Who is the most restricted user of this object?
2. Is there ever going to be an instance of this object that this user shouldn't be allowed to see?
3. Is there ever going to be an instance of this object that this user shouldn't be allowed to edit?

Sharing Model Settings to choose for an object (and how it effects the records on object):

- Private - only record owner and users above via role hierarchy can view, edit, and report on the records.
- Public Read Only - All users can view and report on records, only owner and hierarchy can edit.
- Public Read/Write - All users can view, edit, and report on all records.
- Controlled by Parent - A user can view, edit, or delete a record if the user can perform that same action on the ***record** it belongs to. (*this may be a typo...should this be field/object?)

*When changing org-Wide Sharing Defaults via 'Setup', 'Sharing Settings', Salesforce will automatically run a recalculation on your org to apply changes. This can be lengthy depending on org data size!

*Note that disabling role hierarchy access for objects does not remove access from users with: 'View All', 'Modify All', 'View All Data', and 'Modify All Data' system permissions.

Role Hierarchy creation:
Users can access the data of all the users directly below them in the hierarchy. This hierarchy does NOT have to match your org chart. Each role in the hierarchy just represents a level of access needed by a user or group of users.

- A manager always has access to the same data as their employees regardless of the org-wide default settings.
- Users who tend to need access to the same types of records can be grouped together