Study Guide: Security (10%)

Overview

The Security section of the Salesforce Platform App Builder Certification Exam, weighted at 10%, evaluates your ability to configure and manage access to data and features within a Salesforce org. Security is a critical aspect of any Salesforce implementation, ensuring that users can only see and do what they're authorized to, protecting sensitive data, and maintaining compliance with business policies. This topic is all about declarative tools—no coding required—though understanding the interplay of various security layers is key.

You'll need to master object-level security (via Profiles and Permission Sets), field-level security, record-level security (via Org-Wide Defaults, Role Hierarchy, Sharing Rules, and Manual Sharing), and related concepts like record ownership. This guide will break it all down with in-depth content, real-world scenarios, and study aids to help you excel on the exam and in practical applications.

Key Concepts

Here's what you'll need to master:

1. **Object-Level Security**: Control access to objects using Profiles and Permission Sets.
2. **Field-Level Security (FLS)**: Restrict access to specific fields within objects.
3. **Record-Level Security**: Manage who sees which records via Org-Wide Defaults (OWD), Role Hierarchy, Sharing Rules, and Manual Sharing.
4. **Record Ownership**: Understand how ownership affects access.
5. **Profiles**: Define baseline permissions for users.
6. **Permission Sets**: Grant additional access without changing Profiles.
7. **Role Hierarchy**: Provide vertical access to records.
8. **Sharing Model**: Balance security and collaboration.

Detailed Explanation

1. Object-Level Security

Object-Level Security determines what users can do with entire objects (e.g., Accounts, Opportunities), controlled primarily through Profiles and Permission Sets.

- **Permissions**:
    - **Read**: View records.
    - **Create**: Add new records.
    - **Edit**: Modify records.
    - **Delete**: Remove records.
    - **View All**: See all records (bypasses record-level security).
    - **Modify All**: Edit/delete all records (bypasses record-level security).
- **Example**: Sales reps need to create and edit Opportunities but not delete them:
    - Profile: "Sales User" → Opportunity: Read, Create, Edit (no Delete).
- **Real-World Scenario**: HR users need full access to a custom Employee__c object:
    - Profile: "HR User" → Employee__c: Read, Create, Edit, Delete.
- **Best Practices**:
    - Limit "View All" and "Modify All"—they override record-level security.
    - Use Permission Sets for exceptions rather than creating new Profiles.
    - Regularly audit object permissions for compliance.

2. Field-Level Security (FLS)

Field-Level Security restricts access to specific fields within an object, even if a user can see the record.

- **Settings**:
    - **Visible**: User can see and edit the field.
    - **Read-Only**: User can see but not edit.
    - **Hidden**: Field is invisible.
- **Configuration**: Set in Profile or Permission Set under Field Permissions.
- **Example**: Hide Salary__c on Employee__c from non-HR users:
    - Profile: "Standard User" → Salary__c: Hidden.

    - Profile: “HR User” → Salary__c: Visible, Editable.
- **Real-World Scenario**: Sales reps see but can’t edit Contract_End_Date__c on Account:
    - Profile: “Sales User” → Contract_End_Date__c: Read-Only.
- **Tips**:
    - Apply FLS to sensitive fields (e.g., SSN, financial data).
    - Check FLS when fields don’t appear on layouts—often overlooked.
    - Use Permission Sets to grant field access to specific users.

3. Record-Level Security

Record-Level Security controls which records a user can see or edit, layered on top of object-level permissions.

- **Layers**:
    1. **Org-Wide Defaults (OWD)**: Baseline access for all records of an object.
    2. **Role Hierarchy**: Grants access up the chain.
    3. **Sharing Rules**: Extend access based on criteria or roles.
    4. **Manual Sharing**: User-driven exceptions.
- **OWD Options**:
    - **Private**: Only the owner (and admins) can see/edit.
    - **Public Read Only**: All users can see, owners/admins can edit.
    - **Public Read/Write**: All users can see/edit.
- **Example**: Set Opportunity OWD to Private:
    - Only owners see their Opportunities unless sharing is added.
- **Real-World Scenario**: Customer data (Account) is sensitive:
    - OWD: Private.
    - Sharing Rule: Share with “Sales Team” role if Type = "Customer".

4. Record Ownership

Every record has an owner (usually a user, sometimes a queue), which impacts access.

- **Default Behavior**:
    - Owners have full access (Read, Edit, Delete) unless restricted by Profile.
    - OWD determines if others see the record.
- **Example**: A rep owns an Opportunity:
    - OWD: Private → Only the rep sees it.
    - Role Hierarchy: Manager above rep sees it too.
- **Real-World Scenario**: Support Cases in a queue:
    - Queue owns unassigned Cases → Queue members can take ownership.
- **Tips**:
    - Use queues for shared ownership (e.g., "Support Queue").
    - Transfer ownership via automation (e.g., Flow) for process efficiency.

5. Profiles

Profiles are the foundation of user permissions, defining baseline access to objects, fields, and features.

- **Characteristics**:
    - One Profile per user.
    - Controls object permissions, FLS, app visibility, and more.
- **Standard Profiles**: System Administrator, Standard User, etc.
- **Custom Profiles**: Clone and customize (e.g., "Sales Rep Profile").
- **Example**: "Marketing User" Profile:
    - Campaign: Read, Create, Edit.
    - Lead: Read, Create.
- **Real-World Scenario**: Restrict service reps:
    - Profile: "Service User" → Opportunity: No access.
- **Best Practices**:
    - Keep Profiles broad—use Permission Sets for specifics.

    - Clone standard Profiles to save time.
    - Avoid over-customization—leads to maintenance headaches.

6. Permission Sets

Permission Sets grant additional access without altering a user's Profile.

- **Use Cases**:
    - Temporary access (e.g., project-specific permissions).
    - Role-based exceptions (e.g., "Delete" for a few users).
- **Example**: Grant "Delete" on Case to select reps:
    - Permission Set: "Case Deleter" → Case: Delete.
    - Assign to specific users.
- **Real-World Scenario**: Auditors need Employee__c access:
    - Permission Set: "Audit Access" → Employee__c: Read.
- **Tips**:
    - Stack multiple Permission Sets per user.
    - Name clearly (e.g., "Opportunity_Edit_Override").
    - Use for field-level exceptions too.

7. Role Hierarchy

Role Hierarchy provides vertical access, letting users higher in the chain see/edit subordinates' records.

- **Behavior**:
    - Inherits OWD access (e.g., Private → managers see subordinates' records).
    - Does not grant object-level permissions—Profile must allow access.
- **Example**: Sales org:
    - Roles: CEO → VP Sales → Sales Manager → Sales Rep.
    - OWD: Private → Sales Manager sees all Reps' Opportunities.
- **Real-World Scenario**: Regional support:

    - Roles: Global Support → NA Manager → NA Rep.
    - NA Manager sees NA Reps' Cases.
- **Best Practices**:
    - Align with org chart.
    - Avoid flat hierarchies—limits sharing flexibility.
    - Test with sample records.

8. Sharing Rules

Sharing Rules extend record access beyond OWD and Role Hierarchy based on criteria or roles.

- **Types**:
    - **Owner-Based**: Share records owned by a role/group with another role/group.
    - **Criteria-Based**: Share records meeting conditions (e.g., Amount > 10000).
- **Access Levels**: Read Only, Read/Write.
- **Example**: Share high-value Opportunities:
    - Criteria: Amount > 50000.
    - Share with: "Finance Team" → Read Only.
- **Real-World Scenario**: Cross-team collaboration:
    - Owner-Based: Share "Sales Reps" Accounts with "Support Reps" → Read Only.
- **Tips**:
    - Use sparingly—too many rules slow performance.
    - Test with users in different roles.
    - Combine with Role Hierarchy for layered access.

Study Guide Tables

Table 1: Security Layers

Layer	Scope	Tool	Example
Object-Level	Entire objects	Profiles, Permission Sets	Allow Edit on Opportunity
Field-Level	Specific fields	FLS	Hide Salary__c
Record-Level	Individual records	OWD, Sharing Rules	Private Accounts

Table 2: OWD Options

Setting	Access Level	Use Case
Private	Owner only	Sensitive data
Public Read Only	All see, owners edit	Reference data
Public Read/Write	All see/edit	Collaborative data

Table 3: Profiles vs. Permission Sets

Feature	Profile	Permission Set
Assignment	One per user	Multiple per user
Scope	Broad baseline	Specific overrides
Example	“Sales User”	“Case Delete Access”

Practical Examples

1. **Object-Level**: Restrict Lead deletion:
    - Profile: “Marketing User” → Lead: Read, Create, Edit (no Delete).
2. **FLS**: Hide SSN__c on Contact:
    - Profile: “Standard User” → SSN__c: Hidden.
3. **Record-Level**: Private Case with sharing:
    - OWD: Private.
    - Sharing Rule: Share with “Support Team” if Priority = "High".
4. **Role Hierarchy**: Sales access:

    - Role: Sales Manager → Sees all Reps' Opportunities.

Tips for Success

- **Hands-On Practice**: In a Developer org:
    - Create a Profile with limited Opportunity access.
    - Set up FLS to hide a field on Account.
    - Configure OWD Private and a Sharing Rule for Case.
- **Layered Approach**: Understand how security builds (OWD → Role → Sharing).
- **Trailhead Modules**:
    - "Data Security"
    - "User Authentication"
- **Scenarios**: Practice questions like:
    - "Restrict X users from Y data."
    - "Grant Z access to W records."
- **Test with Users**: Log in as different Profiles to verify access.
- **Audit Tools**: Use Security Health Check in Setup.
- **Exam Traps**: Watch for "least privilege" answers—Salesforce favors minimal access.

Bullet Point Summary

- **Object-Level Security**:
    - Controlled by Profiles/Sets.
    - Define CRUD permissions.
    - Limit "View All."
- **Field-Level Security**:
    - Hide or restrict fields.

    - Set in Profiles/Sets.
    - Protect sensitive data.
- **Record-Level Security**:
    - OWD sets the baseline.
    - Role Hierarchy extends up.
    - Sharing Rules broaden access.
- **Record Ownership**:
    - Owners get full access.
    - Use queues for teams.
    - Transfer via automation.
- **Profiles**:
    - One per user.
    - Broad permissions.
    - Clone to customize.
- **Permission Sets**:
    - Add specific access.
    - Stackable.
    - Flexible exceptions.
- **Role Hierarchy**:
    - Vertical access.
    - Align with org.
    - Test with OWD.

Study Plan

1. **Day 1**: Study Object-Level and FLS (configure a Profile).
2. **Day 2**: Master OWD and Role Hierarchy (set up a hierarchy).

3. **Day 3**: Practice Sharing Rules and Ownership (create 2 rules).
4. **Day 4**: Explore Permission Sets (grant exceptions).
5. **Day 5**: Review all layers, test a full scenario.
6. **Day 6**: Take a practice quiz on Security.

Exam Scenarios to Practice

1. **Scenario**: “Only HR sees Salary__c on Employee__c.”
    - Solution: FLS → Hide for non-HR Profiles.
2. **Scenario**: “Sales Managers see all Reps’ Opportunities.”
    - Solution: OWD Private, Role Hierarchy with Managers above Reps.
3. **Scenario**: “Share high-priority Cases with Support Leads.”
    - Solution: Sharing Rule → Priority = "High", Share with “Support Leads.”