CS-063 · Application Security

Authorization

Deciding what a verified identity is allowed to do and access.

Executive Summary

Authorization is the process of deciding what a verified identity is permitted to do and which resources it may access. It comes after authentication, which confirms identity, and it is the control that enforces least privilege. Broken access control, where authorization is missing or wrong, is one of the most common and damaging web application flaws, so getting authorization right is a core security responsibility.

What It Is

Authorization defines and enforces the boundaries of what each user, service, or role can do. Where authentication proves who you are, authorization determines whether you may view a record, edit a setting, or perform an action. It is commonly implemented through models such as role-based access control, where permissions attach to roles, and attribute-based access control, where decisions consider attributes of the user, resource, and context. Good authorization is enforced on the server for every sensitive action and every object, not just hidden in the interface, because anything the client controls can be manipulated by an attacker.

Why It Matters

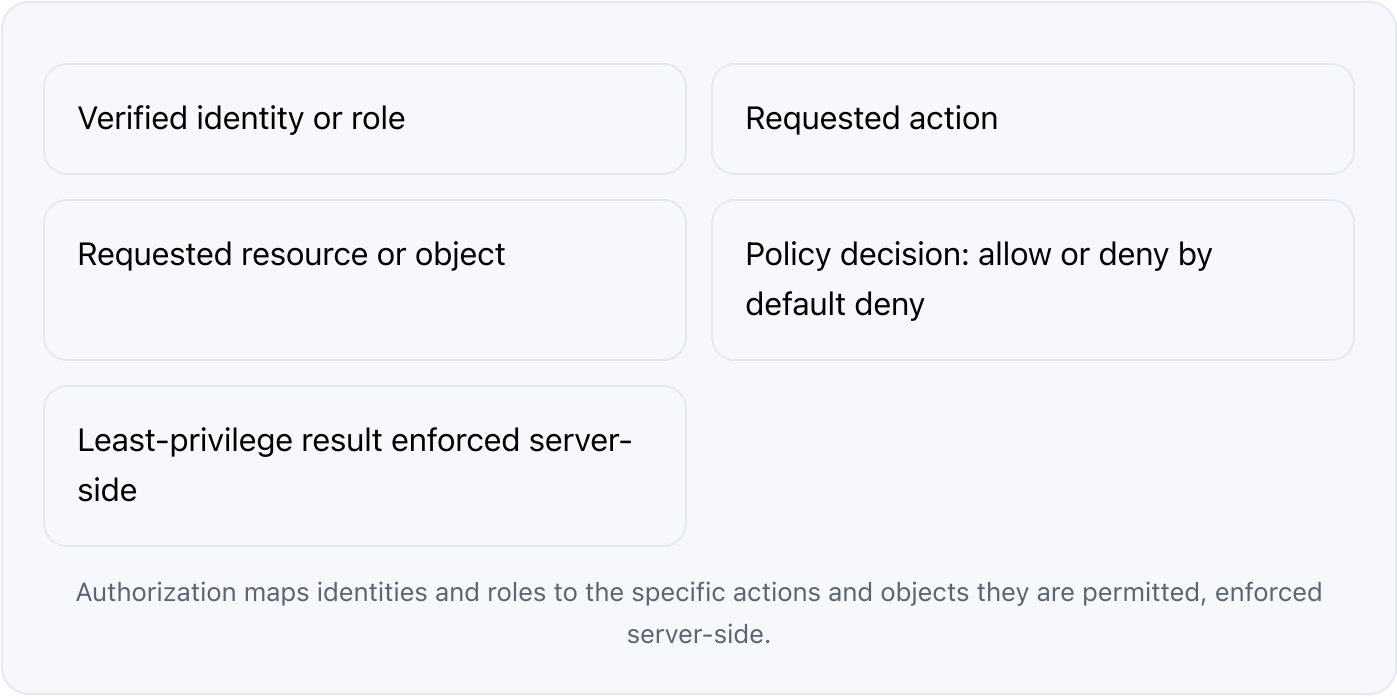
When authorization fails, users can reach data and actions that should be off limits, which is exactly what attackers look for. Broken access control consistently ranks among the most serious web application risks because the impact is direct: one user reading or changing another user's data, a normal user performing administrative actions, or an unauthenticated request reaching protected functionality. These flaws are often simple to exploit and hard to detect from the outside, so building authorization checks deliberately and testing them thoroughly is

essential. For professionals, authorization is a frequent source of real-world breaches and a heavily tested skill.

How It Works

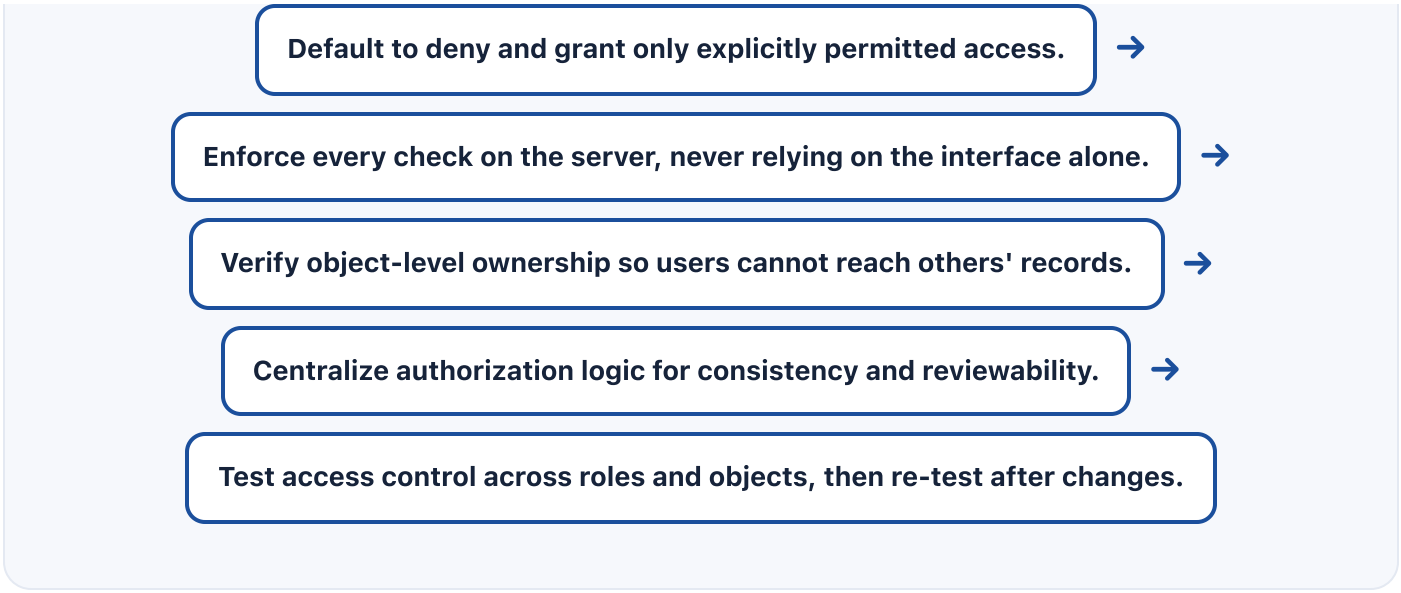
At a conceptual level, authorization is a decision made at the moment of every sensitive request: given this verified identity, is this specific action on this specific resource allowed. The safe default is to deny and to grant only what is explicitly permitted, applying least privilege so each identity has the minimum access it needs. Checks must happen on the server for every request, including checks that the current user actually owns or may access the specific object being requested, which prevents a user from changing an identifier to reach someone else's data. Centralizing authorization logic, rather than scattering ad hoc checks, makes it consistent and reviewable. Sensitive actions and administrative functions deserve extra scrutiny and, where appropriate, additional verification.

Architecture Diagram



Authorization maps identities and roles to the specific actions and objects they are permitted, enforced server-side.

Visual Workflow



Common Attacks

- Changing an identifier to access another user's record, an object-level flaw
- Reaching administrative functions as an ordinary user through missing checks
- Accessing protected functionality without being authenticated at all
- Escalating privileges by manipulating role or permission values
- Bypassing checks that were enforced only in the client interface

Common Mistakes

- Enforcing access control in the interface but not on the server
- Checking that a user is logged in but not that they may access the object
- Defaulting to allow instead of deny for new or undefined cases
- Granting broad roles that far exceed what a job actually needs
- Scattering inconsistent, ad hoc checks that are easy to miss

Best Practices

- Apply least privilege and a default-deny posture everywhere
- Enforce authorization on the server for every sensitive request
- Verify object-level ownership and access on each request
- Centralize authorization decisions rather than duplicating ad hoc checks

- Use a clear model such as role-based or attribute-based access control
- Review and re-certify access periodically to remove creep

Quick Checklist

- [x] Default-deny policy with explicitly granted permissions
- [x] Server-side authorization checks on every sensitive action
- [x] Object-level ownership verified, preventing identifier tampering
- [x] Roles scoped to least privilege, no broad catch-all grants
- [x] Authorization logic centralized and reviewable
- [x] Access control tested across roles and objects before release

Recommended Tools

Policy engine or authorization service
Centralizes and evaluates access decisions consistently

Identity and access management platform
Manages roles, permissions, and access reviews

Dynamic application security testing (DAST)
Tests for missing or bypassable access control on a running app

Access review and certification tooling
Periodically confirms that granted access is still appropriate

Industry Standards

OWASP Top 10
Broken access control is a leading web application risk category

OWASP Application Security Verification Standard (ASVS)
Testable requirements for access control enforcement

NIST SP 800-53
Access control family covering least privilege and separation of duties

Career Relevance

Authorization is core to application security engineers, penetration testers, secure code reviewers, and identity and access management specialists, who focus heavily on access control because broken authorization is so common. Developers must enforce it correctly on the server, and GRC and audit teams assess least privilege and access reviews. For the AI-Governance-Jobs.com audience, object-level authorization and least privilege are frequent, high-signal interview topics.

Interview Questions

- How does authorization differ from authentication?
- What is broken access control, and why is it so impactful?
- What is an object-level access flaw, and how do you prevent it?
- Why must authorization be enforced on the server rather than the client?
- Compare role-based and attribute-based access control.

Related Certifications

CompTIA Security+ **ISC2 Certified in Cybersecurity (CC)** **ISC2 CISSP (for leadership tracks)**

Further Reading

- OWASP: Broken Access Control
- NIST SP 800-53 Access Control Family
- CWE-285: Improper Authorization

Key Takeaways

- Authorization decides what a verified identity is allowed to do.
- Broken access control is among the most common and damaging flaws.
- Default to deny and apply least privilege everywhere.
- Enforce checks on the server, including object-level ownership.
- Centralized, tested authorization logic is far safer than ad hoc checks.

FAQ

▶ **How is authorization different from authentication?**

▶ **What is an object-level access flaw?**

▶ **Why does client-side authorization fail?**

Related Careers

RELATED ROLES

Application Security Engineer
Penetration Tester
Secure Code Reviewer
Security Engineer

RELATED CERTIFICATIONS

CompTIA Security+
ISC2 Certified in Cybersecurity (CC)
ISC2 CISSP (for leadership tracks)

CURRENT OPENINGS

Live roles are on the job board.

Browse all jobs

GUIDES & SALARY

Career guides
Role roadmaps
Salary data
Career tools

SUGGESTED LEARNING PATH

1 Ground the basics with CS-001 Cybersecurity

2. Study this sheet: **Authorization**
3. Go deeper: Authentication
4. Go deeper: Cross-Site Request Forgery (CSRF)
5. Validate it: work toward **CompTIA Security+**
6. Find the role: browse current openings

RELATED SHEETS

CS-062 Authentication | CS-060 Cross-Site Request Forgery (CSRF) | CS-057 OWASP Top 10 | CS-064 Secrets Management | CS-001 Cybersecurity