# Broken Function Level Authorization

occurs when backend functionalities (like APIs or routes) fail to enforce proper user permissions, allowing unauthorized users to access or invoke functions intended only for privileged users.

## Example:

Regular user sending   
POST /admin/deleteUser

Even though there's no admin panel UI, the backend function is accessible if the request is crafted manually.

## Mitigation:

Role-based access checks on each function, not just UI.

# Difference Between Broken Access Control vs. Broken Function-Level Authorization

| Broken Access Control | Broken Function-Level Auth |
| --- | --- |
| Focuses on data or object access | Focuses on functionality or actions |
| Example: View another user’s invoice | Example: Perform admin action (e.g., delete user) |

# Impact

| Type | Description |
| --- | --- |
| Technical | Unauthorized function execution (delete, create, update sensitive data). |
| Security | Users gain admin or elevated privileges silently. |
| Business | Data manipulation, privilege escalation, compliance risks (GDPR, HIPAA), brand damage. |

# Common Scenarios

| Scenario | Description |
| --- | --- |
| Admin-only API endpoints not properly secured | e.g., /admin/createUser, /internal/systemLogs |
| UI hides admin functionality, but backend doesn’t enforce it | Relying on front-end only for access restriction |
| Misconfigured HTTP methods allowed | e.g., PUT, DELETE, POST used without verification |
| Mobile app or API calls reveal admin features | Hidden actions exposed via API proxy tools (Burp/Postman) |

# Testing Methods (Red Team & Pentester View)

| Technique | Tool | Use |
| --- | --- | --- |
| Force-browsing to hidden APIs | Burp Suite → Repeater | Manually test endpoints |
| Role abuse testing | AuthMatrix (Burp Plugin) | Test if lower role can invoke higher functions |
| Method tampering | Postman / Burp | Change GET to PUT, POST, etc. |
| Parameter Pollution | Burp Intruder | Add extra parameters to call protected logic |

# Mitigation Strategies

| Area | Control |
| --- | --- |
| Backend Authorization Checks | Always enforce authorization on the server side. |
| Role-Based Access Control (RBAC) | Backend must validate roles before performing sensitive functions. |
| Centralized Access Control Layer | Avoid scattered permission logic in the codebase. |
| Never Trust the UI | Hidden buttons/menus don't equal security — backend must enforce checks. |
| HTTP Method Restrictions | Use web server/WAF to block unused HTTP methods (e.g., disable PUT if not required). |
| Logging + Monitoring | Alert on privilege violations or unauthorized function attempts. |

# Type-wise Classification (Under Broken Function-Level Auth)

| Type | Definition | Example | Mitigation |
| --- | --- | --- | --- |
| 1. Hidden Admin APIs | Functionality accessible if guessed or discovered | /admin/exportLogs | Auth check in backend |
| 2. UI Bypass / Force Browsing | No frontend link, but endpoint still accessible | Manual POST to /system/reset | RBAC enforcement |
| 3. HTTP Method Exploits | Using unintended methods | PUT /user/1000/upgradeToAdmin | Restrict & verify methods |
| 4. Client-Side Role Control | Role info stored/verified in client-side JS | localStorage.setItem('role', 'admin') | Never trust client |

# Tools to Detect It

| Tool | Purpose |
| --- | --- |
| Burp Suite + Repeater | Manually modify API routes or parameters |
| Postman | Test various API methods and role access |
| OWASP ZAP | API fuzzing (scripted) |
| AuthMatrix | Automate role-based function access testing |
| SIEM Logs | Detect abnormal function usage by user role |

# Points

Broken Function-Level Authorization is one of the top API-related issues in modern apps — especially REST APIs and mobile apps."

"Always ensure that developers understand the principle of: 'The backend is the source of truth'."

"We use AuthMatrix in conjunction with Burp Suite to test multi-role access scenarios for web and mobile APIs."