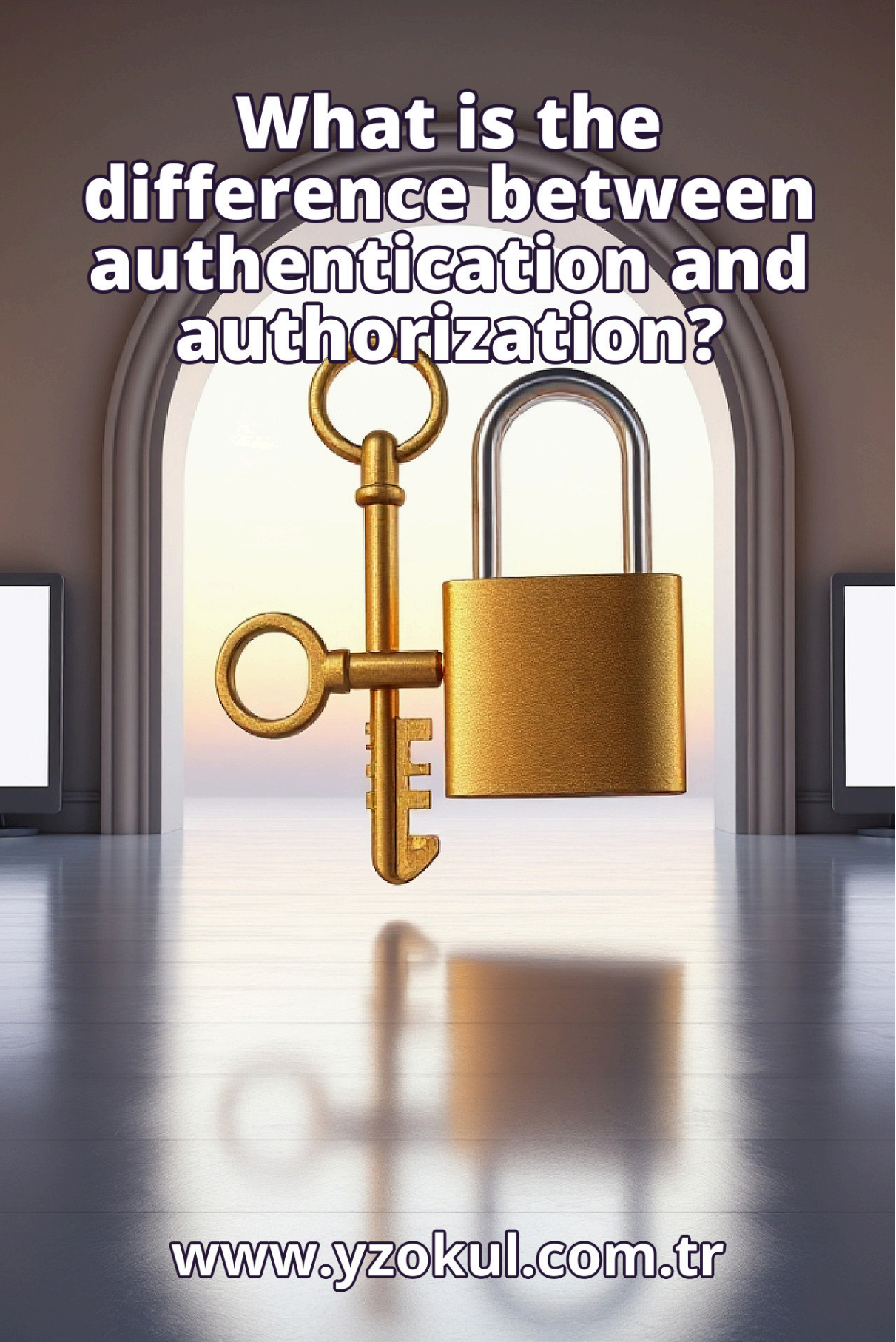
What is the difference between authentication and authorization?



www.yzokul.com.tr

What is the difference between authentication and authorization?



You are asking about authentication and authorization, two words often confused. The short answer is simple: authentication checks who you are, and authorization checks what you may do. Keep this one-line answer in mind as you read the details. It gives you the whole idea before you look closer. Many beginners mix them up, but they are separate ideas.



Two different questions stand behind these two words. Authentication answers the question who are you, while authorization answers the question what are you allowed to do. They are related because a system usually needs to know who you are before it can decide what you may do. But they are not the same thing. Treating them as one leads to confusion, so keep the two questions apart.



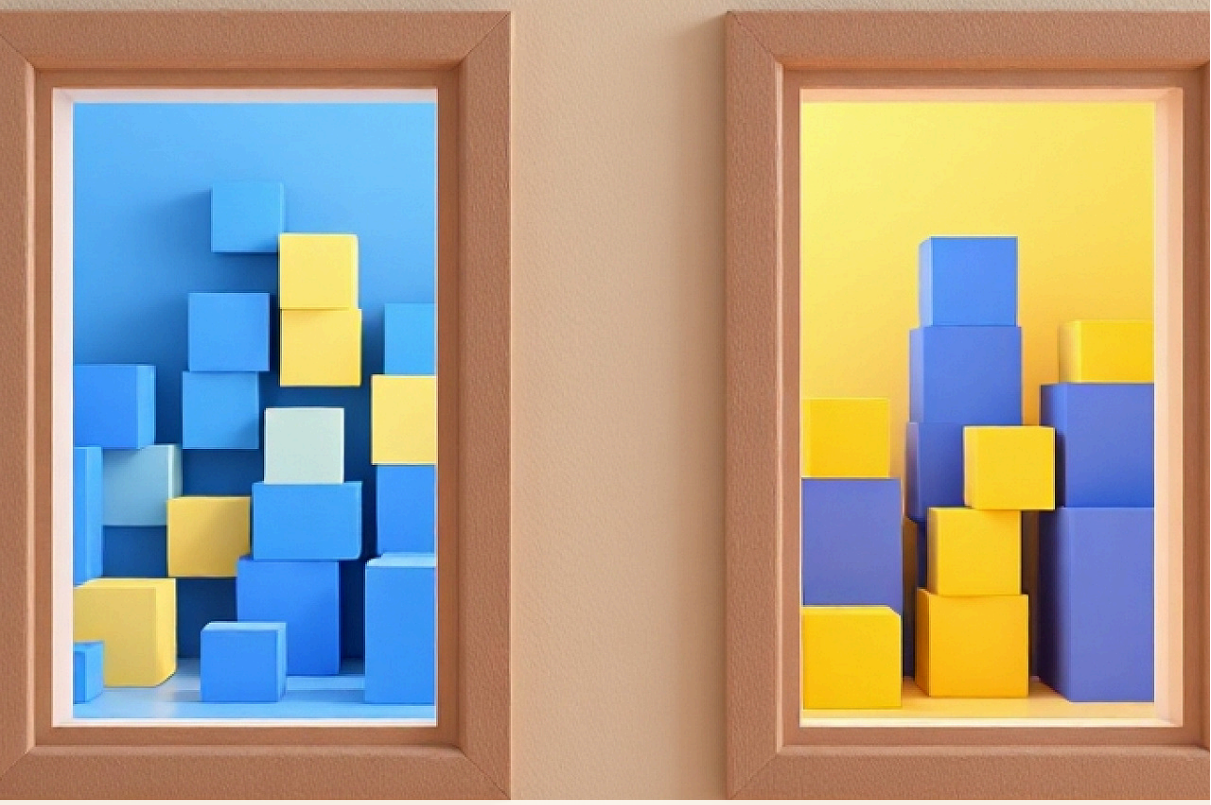
Authentication means proving your identity to a system. It is the act of showing that you really are the person you claim to be. A system uses authentication to be sure about who is on the other side. Without it, the system cannot tell you apart from anyone else. So authentication is about identity, not about permissions or access rights.



A login screen is where authentication usually happens. You type a username and a password into the form, and the system compares them with the stored values it already has. If the two match, the system accepts that you are the person you claim to be. If they do not match, access is refused. This check is the core of authentication.



Authorization is the decision about which actions and data are allowed for a given user. It happens after the system knows who you are. The system looks at your identity and your role, then decides what you may do. Authorization is about permissions and access rights, not about identity. It answers what you may do, given who you are.



Picture a school system where students and teachers both log in. Everyone uses the same login screen and the same kind of username and password. Authentication treats them the same way and confirms each person's identity. But inside the system, students and teachers are allowed to do different things. That difference is where authorization comes in.



Follow a student who logs in to the school system. Authentication succeeds because the username and password match the stored values. The student can then view their own grades on a page. This shows that authentication worked and a limited permission was granted. The student sees only their own grades, not everyone else's.



Now the same student tries to change a grade. The system knows exactly who the student is, so authentication has already passed. But editing grades is not allowed for students, so the system refuses the action. This refusal is an authorization decision. Being recognized is not the same as being allowed to do everything.



Compare this with a teacher who logs in to the same school system. The teacher types a username and password, and authentication succeeds in the same way. But the teacher is allowed to edit grades, because the role is different. Same login step, different permissions. Authorization gives the teacher rights that the student does not have.



A successful login does not open all data. Being recognized by the system is not the same as being trusted with everything inside it. The student who logs in cannot see other students' grades or change any grade. So authentication alone is not enough. The system must still decide what each recognized user may do.



The application must check permissions on every important operation, not just once at login. Each sensitive action needs its own authorization check before it runs. For example, editing a grade must be checked every time someone tries it. If the check is skipped, the wrong user could do the wrong thing. Checking once at login leaves later actions unprotected.



Notice the order of the two steps. First the system confirms who you are through authentication. Then it decides what you may do through authorization. Authorization needs a known user to make sense, because permissions are tied to a specific identity. Without a confirmed identity, the system cannot decide which rights apply.



A common mistake is using authentication and authorization as if they were the same word. People say a user is authenticated when they really mean the user is allowed to do something. Keeping the two words apart prevents confused thinking about access. When you separate identity from permissions, your design becomes clearer and safer.



Another common mistake is trusting the login too much. Some assume that a logged-in user should reach everything in the system. The school example shows why that assumption is wrong and unsafe. A logged-in student still must not edit grades. Login proves identity, but it never means full access to all data.



Forgetting the permission check on an important operation is a serious mistake. If the system only checks who you are but never checks what you may do, the wrong user can perform the wrong action. Every important action needs its own authorization check. Skipping it leaves a gap that anyone who logs in could walk through.



Remember the core difference: authentication is who you are, and authorization is what you may do. A login never means full access to all data. The system must check permissions on every important operation, not just once. Keep these points in mind, and you will avoid the most common access mistakes.