Framework Route Authorization

Java Accelerator 7

Lesson 6.4

WELCOME

Learning Outcomes

By the end of this lesson, you will be able to:

Use Spring Security to secure a cloud-native Java application.

Spring Security

Spring Security

Our team developed an Event RSVP feature for our company's conference app. Before we deploy it, we need to add some basic security features.

Spring Security

We want to be able to do the following:

- Display public and private events.
- Allow all users to see and register for all public events.
- Allow only registered users to see and register for private events.
- Allow admins to assign event publisher roles to users.
- Give publishers access to create, update, publish, and unpublish events.

Event RSVP Feature

	All Users	Registered Users	Admins	Publisher Access
Register for all public events	✓	✓	✓	✓
Register for all private events	✗	✓	✓	✓
Assign event publisher roles to users	✗	✗	✓	✗
Give the publishers access to view a guest list	✗	✗	✓	✗
Create, update, and publish and unpublish events	✗	✗	✓	✓
Delete events	✗	✗	✓	✗

Event RSVP Feature

The Problem: Some overlaps exist in the roles for the Event RSVP application. We need to think this through before we implement a solution.

	All Users	Registered Users	Admins	Publisher Access
Register for all public events	✓	✓	✓	✓
Register for all private events	✗	✓	✓	✓
Assign event publisher roles to users	✗	✗	✓	✗
Give the publishers access to view a guest list	✗	✗	✓	✗
Create, update, and publish and unpublish events	✗	✗	✓	✓
Delete events	✗	✗	✓	✗

How is access controlled to a multi-tenant office building?

How could someone break into the different parts of the building?

Security Basics

Authentication

Are you **authentically** you? Are you who you claim to be?

How do we get into different parts of a building?

- Keys
- Keypad entry
- Thumbprint authentication
- Retina scan
- Credential-based authentication

Authentication refers to positively identifying a user—in other words, making sure a user is who they say they are.

Authentication

Computer/web-based authentication is just an extension of something we do in the analog world.

- We have government-issued picture IDs.
- We can be positively identified by our fingerprints or by retina scans.
- We are asked to present an ID when cashing checks, checking into a hotel, or flying on commercial airlines.

Authorization is all about who can do what, sometimes known as **permissions**.

Authorization assumes that authentication has worked and we know who the user is.

Permissions

Permissions are generally not applied directly to users.
Permissions are granted to groups or roles.

	All Users	Registered Users	Admins	Publisher Access
Register for all public events	✓	✓	✓	✓
Register for all private events	✗	✓	✓	✓
Assign event publisher roles to users	✗	✗	✓	✗
Give the publishers access to view a guest list	✗	✗	✓	✗
Create, update, and publish and unpublish events	✗	✗	✓	✓
Delete events	✗	✗	✓	✗

Users are assigned to groups or roles and assume the permissions of the groups or roles they are in. This is referred to as **role-based access control (RBAC)**.

Time to Code

Secure RSVP

Suggested Time:

Time to Code

Register Endpoint

Suggested Time:

Spring Security Concepts

User

A **user** is someone who uses the system in question.

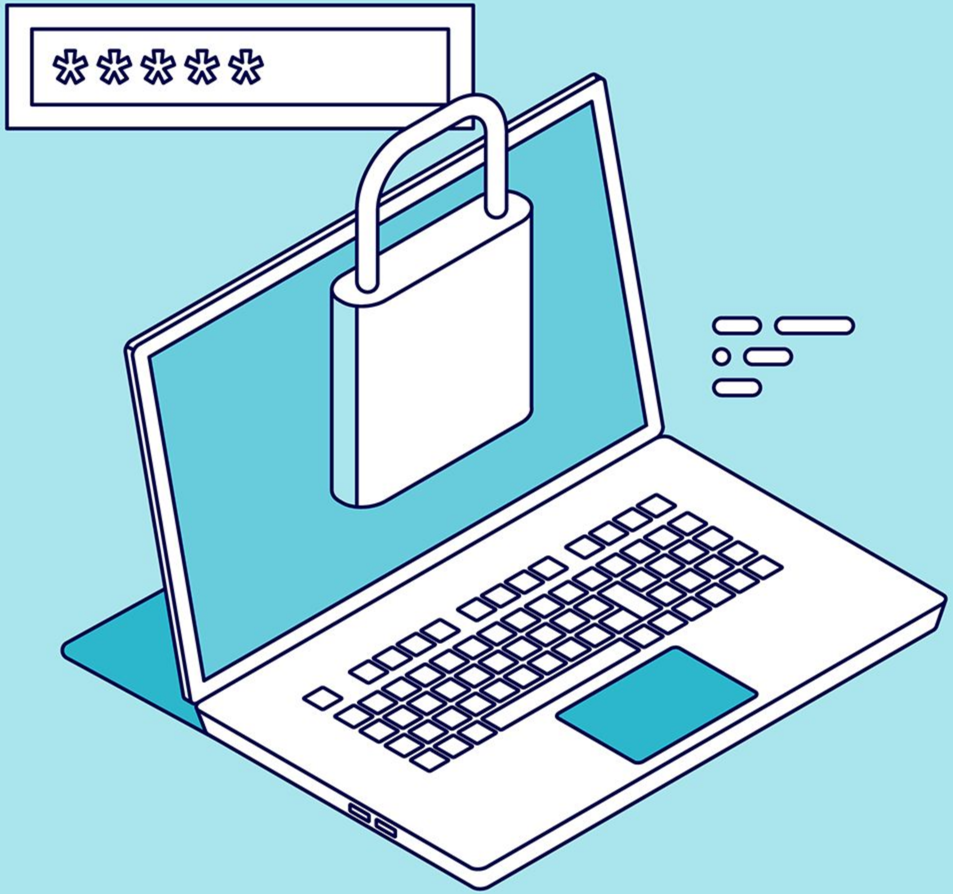
	All Users	Registered Users	Admins	Publisher Access
Register for all public events	✓	✓	✓	✓
Register for all private events	✗	✓	✓	✓
Assign event publisher roles to users	✗	✗	✓	✗
Give the publishers access to view a guest list	✗	✗	✓	✗
Create, update, and publish and unpublish events	✗	✗	✓	✓
Delete events	✗	✗	✓	✗

Users

In Spring, a **user** refers to the account associated with the person who uses the system.

User accounts are generally identified by unique usernames.

The most common way to authenticate a user account is via password.

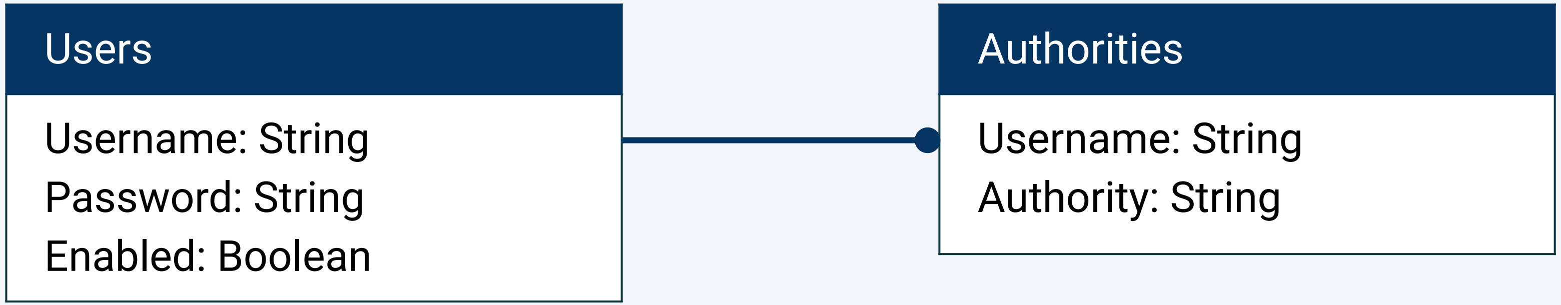


Spring uses the term **authorities** to describe the roles or groups to which user can belong.

Authorities

In Spring security, various authorities are granted to users.

Authorities are arbitrary strings that have no inherent meaning. We get to define the authorities for our applications, and we get to assign meaning to these authorities as we wish.

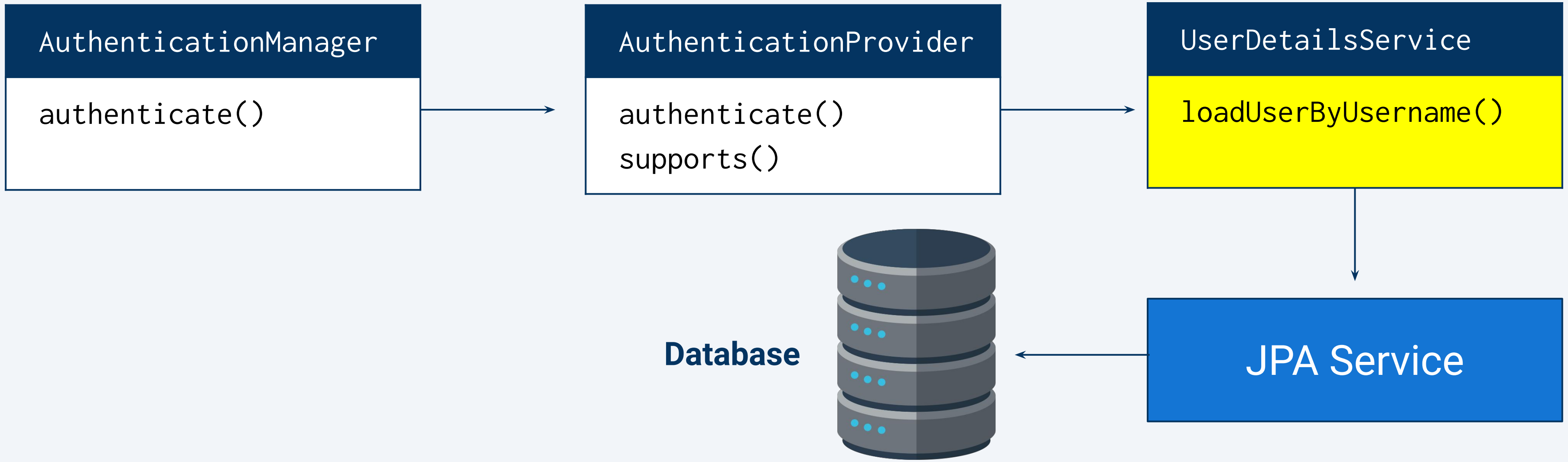


The `UserDetailsService` is the Spring Security component responsible for retrieving information about logged-in users.

User Details Service

We can create custom `UserDetailsService` components to fit whatever user repository the application uses.

Spring Security does have a default implementation if you use a standard user store and access it in a standard way.

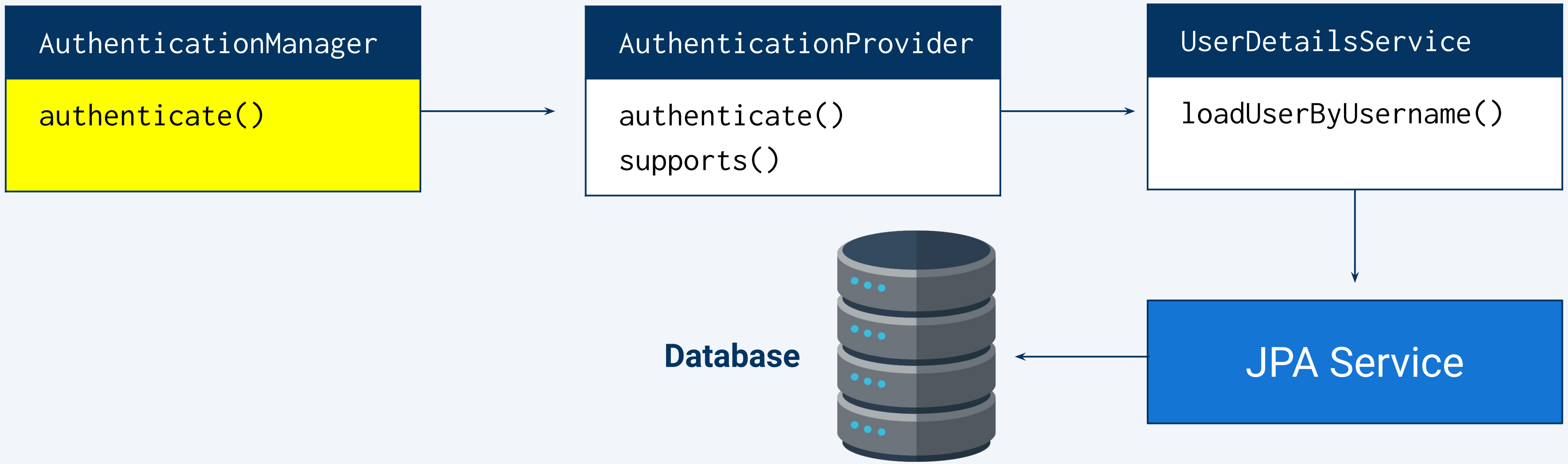


The `AuthenticationManager` component is responsible for authenticating users in the system.

AuthenticationManager

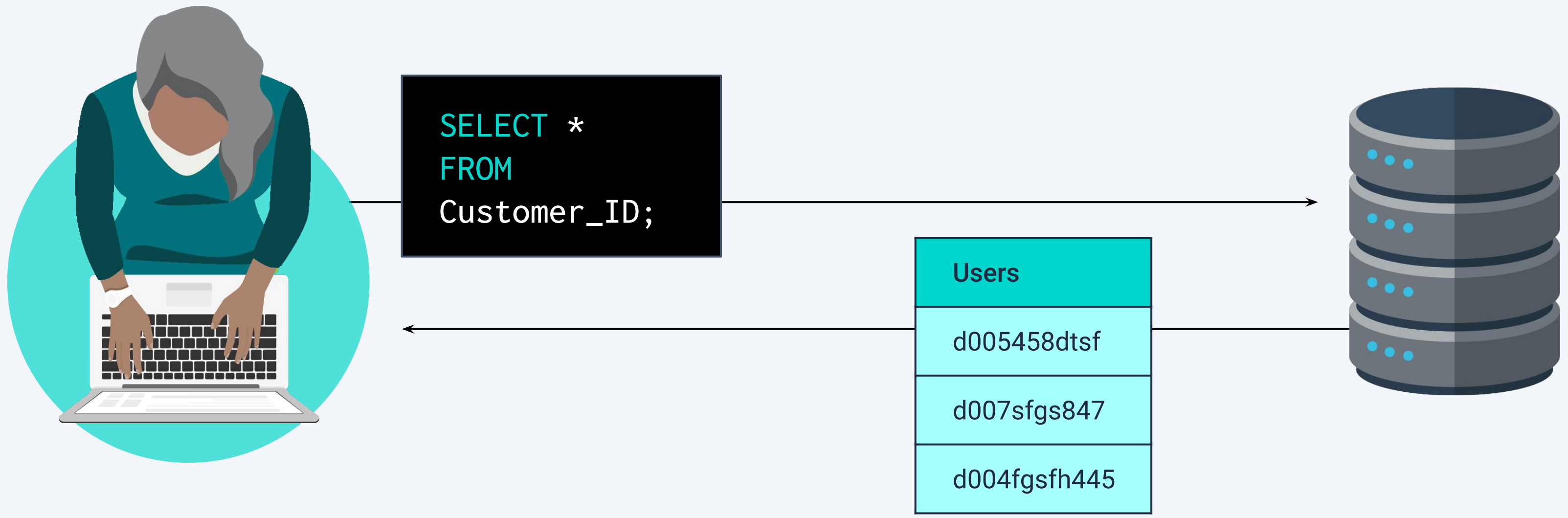
Integrates with whatever user repository the application uses; can find users by username and find their associated authorities.

Spring has some default implementations for common user repositories.



Spring Security

Spring Security supplies a standard SQL schema that can be used for user and authority data. This is the schema we've defined already.



```
SELECT *
FROM
Customer_ID;
```

Users
d005458dtsf
d007sfgs847
d004fgsfh445

Spring Security

Reasons to use Spring Security schema:

01

Spring Security works well for projects that are being built from scratch or that don't already have a user repository.

02

Spring Security makes it easy to incorporate this schema into the `AuthenticationManager` for your application.

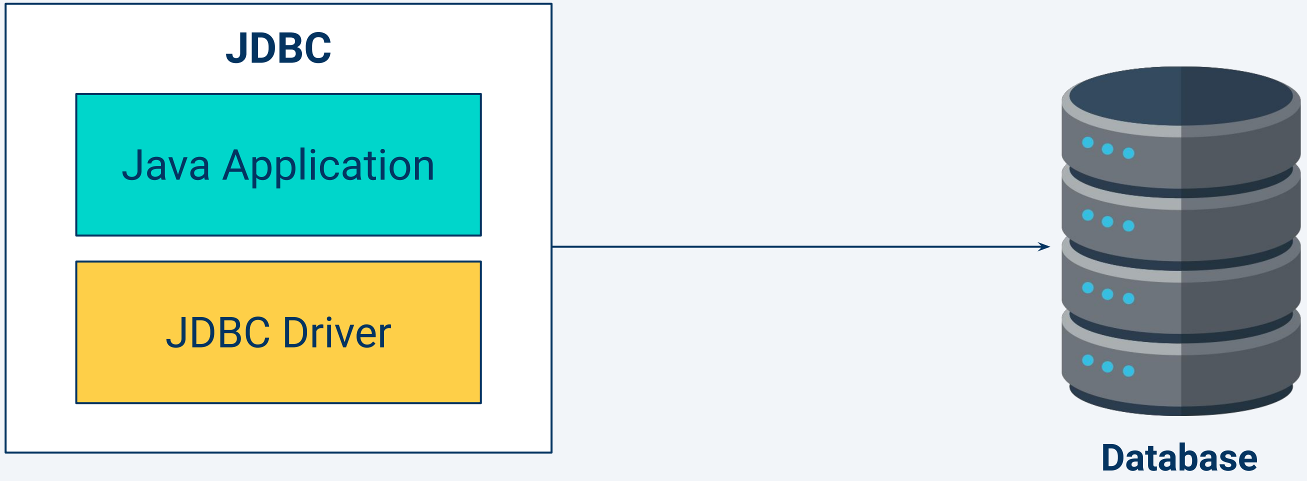
03

Spring Security is a great choice if you don't already have a user/authority store.

Data Source

This provides database connection configuration if you choose to use the Spring Security default SQL schema.

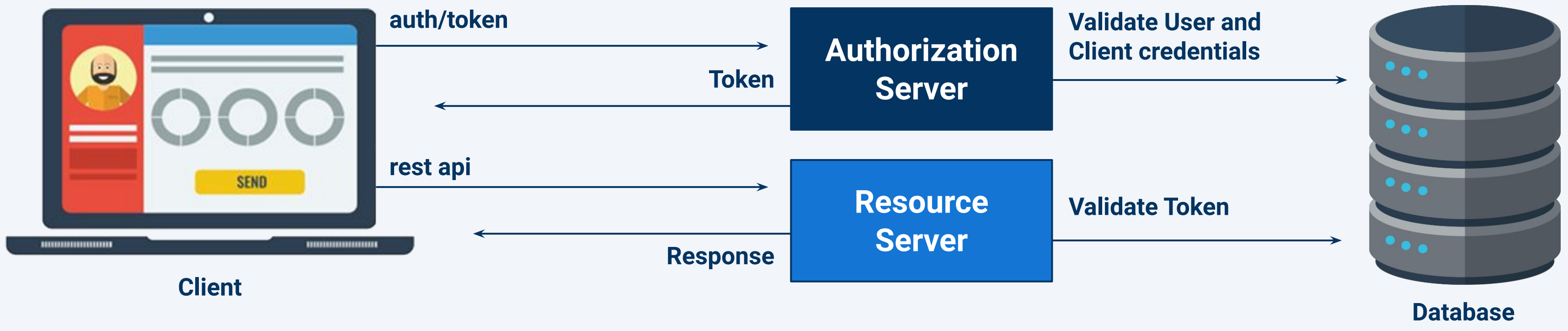
This is the same data source used for general JDBC database connectivity.



Spring Security applies authorization rules to individual or groups of endpoints.

Endpoint Authorization

This is done by defining which authorities (typically called **roles** in this context) a user must have to access a given endpoint.



Endpoint Authorization

We have the freedom to:

01

Leave some endpoints open to everyone.

02

Leave some endpoints open to all authenticated users.

03

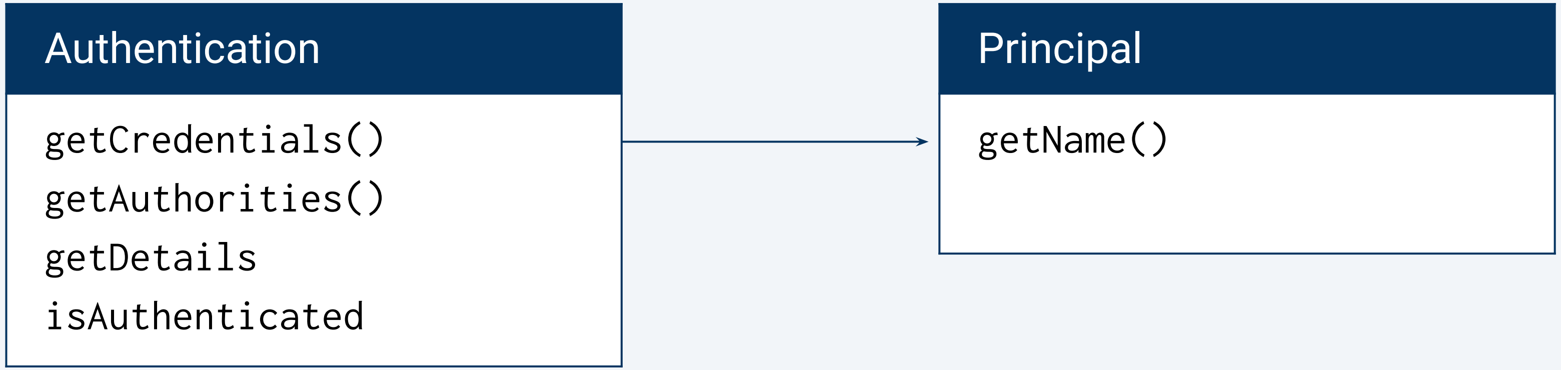
Limit access to some endpoints to only those users who have been granted particular authorities.

The `Principal` is an object that represents a logged-in user of the system.

Principal

The `Principal` includes information such as the user's name and granted authorities.

This is the object we use to get information about the logged-in user in the code.



Logout

Spring Security supports the ability to log users out of the system.

We can define the following:

- The URL to be accessed when requesting to be logged out.
- The URL to which the user should be redirected on successful logout.
- Cookies we want to delete.
- Whether or not the HTTP session should be invalidated on logout.

“**Cross-Site Request Forgery (CSRF)** is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing.”

OWASP®

—The Open Web Application Security Project

Time to <code>