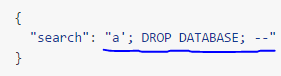
HTTP traffic can be sniffed.. use HTTPs for passwords etc.

Even if HTTPs.. still vulnerabilities:

* SQL Injection
* Cross-site Scripting (XSS)
* Cross-site Request Forgery (CSRF)

If our app contains code which uses **input** from users/clients directly when issuing SQL commands.. our app is vulnerable. 



To ensure this won‘t happen.. all **input must be escaped**.. for instance.. SINGLE QUOTES from users must never be allowed to be interepreted as a part of the SQL command.. and should **always be interpreted as data.**

Our DB connection should be configured such that the DB user doesn‘t have permissions to drop database etc. As well, but doesn‘t solve the whole problem.

**Cross-site scripting**

Apps accept input from users (comment on discussion board f.ex.). Store the input in the DB and return it to other clients.

Malicious user posts: 

And that input is **rendered as HTML**.. it will be EXECUTED on all clients requesting the data..

Important that such **INPUT IS ESCAPED**.. in this case by converting „less-than“ and „bigger-than“ characters into their HTML entities.. < and >.. Markdown becoming popular due to this..

**Cross-site Request Forgery**

Nothing to do with un-encoded data..

* Alice opens website A, for instance a Single-Page Application, which connects to some Web API B
* Alice authenticates via some means. A **cookie** is issued for Alice, and contains either a **token** or some **session ID.** Cookies are issued per domain, in this case.. web.api.for.A.com
* The next time Alice uses A to perform some operation in B (say, /api/courses/{id}/grades – PUT – because Alice is a teacher), the **cookie** is passed with the request. B knows that it is Alice which is performing the action and sees that she is allowed to **update the grades** in the given course.
* Alice opens a malicious website E. Created by Eve.. a student of Alice‘s. This website may be on a different domain than A
* E contains code like this:



Because the PUT request is being made to the domain of the API, and Alice has a valid cookie which the browser sends automatically with each request to the domain.. Eve has thereby tricked Alice to update her grade..

**Best way is to NOT USE COOKIES AT ALL**.. but explicitly include **authentication detials** in the **HTTP header** of each request.

Another option is for the API to have a „whitelist“ of clients which are allowed to perform given/all operations.