Security
Obvious threats -
- Telnet - old "SSH" but not secure. Unencrypted
- FTP - not secure like encrypted SFTP
- HTTP - computational costs of running everything over ssl is still high enough to warrant using http for enough things
- MySQL - not that secure, but has ssl compatibility
    - courses website uses databases on same server, so not as big a deal

suPHP - if generic use of server (serving up static content) using username "root" rather than something else (like "nobody" or "somebody" can make it susceptible to attacks.
- may necessitate permission changes

most processes should not be run as root
- nobody isn't really a perfect solution though

suPHP - substitute user PHP allows you to execute PHP files under the username that owns that particular file
- means if there is a security problem in your code, it only makes your code susceptible

Cookies - the means by which a cookie is sent from server to client is through http headers.

PHPSESSID - represents the session
- name of the file in the temporary directory of the server that just served up the information

PHP - function called serialize() converts object or array to a long string representation. Cookies help remember all metadata useful to reconstruct session next time user visits.

Session Hijacking (scenarios) - taking someone's session id and (using some special software) sending that phpsessid from their computer to the server to try to trick the server into showing the hacker whatever is being shown on the hackee's computer.
Ways to get session id's
- Physical Access - Being on the same machine as the hackee
- Packet Sniffing - Checking out the http traffic
- Sssion Fixation -
- XSS - Cross site scripting - hijack someone's session (steal their id by tricking their browser into executing some javascript code)

Cookies are just a key-value pair. Some php servers, if they notice cookies cannot be sent, automatically invoke passing of information through url (get variables, etc.)

POST information is not sent in headers (not stopped by max length of url - around 2000 characters)

PHP - sessionregeneratekey() - sends a new sessid on every request
    - adversary only has a brief moment of time to use your sessid
    - potential downfall with ajax (asynchronous requests)

A solution? Just encrypting everything?
    - stops packet sniffing
    - but screws up isp caching
    - computational costs of encryption

Encryption causes "randomness" - can the files then be compressed?

SSL is a solution if your host gives you a unique IP
SSL Certificates - range in price (paying for reputation)

SQL Injection Attacks - The user/hacker can induce a self-created query to be passed
    - use mysql_real_escape_string() to check for this.
        - escapes the string in such a way that even if there is malicious intent, nothing will happen

Client-side Attacks - eg. XSS (Cross site scripting)
- Same-origin Policy - content of one site cannot read content of another
    - affects many things
- There are still many attacks
    - Cross-Site Request Forgery (CSRF/XSRF)
        - Bad guy forges request to a website you've already logged into
        - Defenses -
            - CAPTCHA - those images with words you have to type

XSS - tricking a website that a hackee is visiting into delivering the sessid to their own site

DON'T EVER TRUST USER INPUT!
- check a lot of things!
- encode all user input