ECE568 Lecture 14: Web Authentication and Security

David Lie
Department of Electrical and Computer Engineering
University of Toronto

Lecture Outline

- Web Authentication
  - Basic Authentication
  - Cookie-based Authentication
- Cookies
  - Advantages & Pitfalls of cookies
  - Forgeability of cookies
  - Storage of cookies
- HTML intro

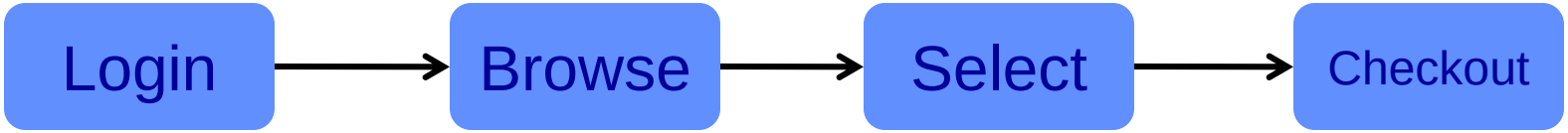
Web Authentication

- A common misconception: using SSL is only part of securing a website. Just because a site uses SSL doesn't mean it's secure!
  - SSL only protects information transferred between the client and server. It provides confidentiality and integrity, but doesn't *authenticate* the person using the client.

- Authentication occurs when you prove to the web server who you are. This is almost always done with a username and password. Without this, the server will not allow certain HTTP requests:
  - You log into your banking website. Without this, the web browser will not let you see the page with your bank account information.

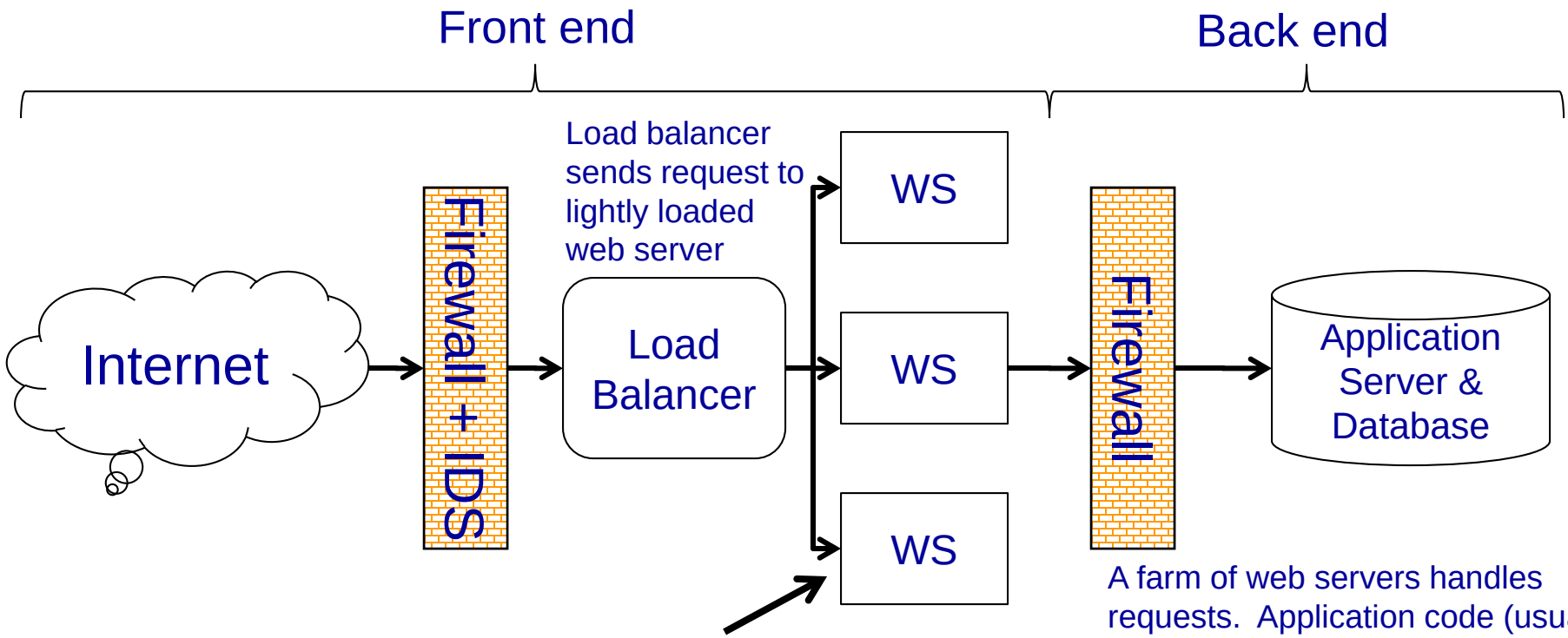
Web Session State

- A web session involves navigating through several pages, each dependent on the previous. This can be called a web session *work flow* or *click-through.* Example:



- Each step depends on the previous step, so the website must "remember" what you have done before to make sure you do not skip any steps:
  - Remembering what has happened in the past requires the website to maintain *state*.
  - The state of your web session dictates where you are in the web site work flow.

Basic web server architecture



Web servers are inherently stateless (they don't remember state). This is to allow easy recovery if a web server fails. All state is stored in the database backend, but access to the database is expensive

A farm of web servers handles requests. Application code (usually in perl/python/Java) handles the request and makes requests/updates to application data bases in the backend.

Web Authentication

- Because the web servers are stateless you need another way to remember if user has logged in or not.
  - Example of a bad solution: remember the IP address the user is at and store that in the database:
  - On every subsequent request the web server checks with the database to make sure the IP address matches
  - This is expensive for the database

- Web authentication instead recruits the help of the client web browser to store the state thus taking load of the database. There are 2 basic methods:
  - Basic web authentication (older, and should not really be used anymore)
  - Cookie-based authentication.

Basic Web Authentication

http://**username**:**password**@www.website.com/

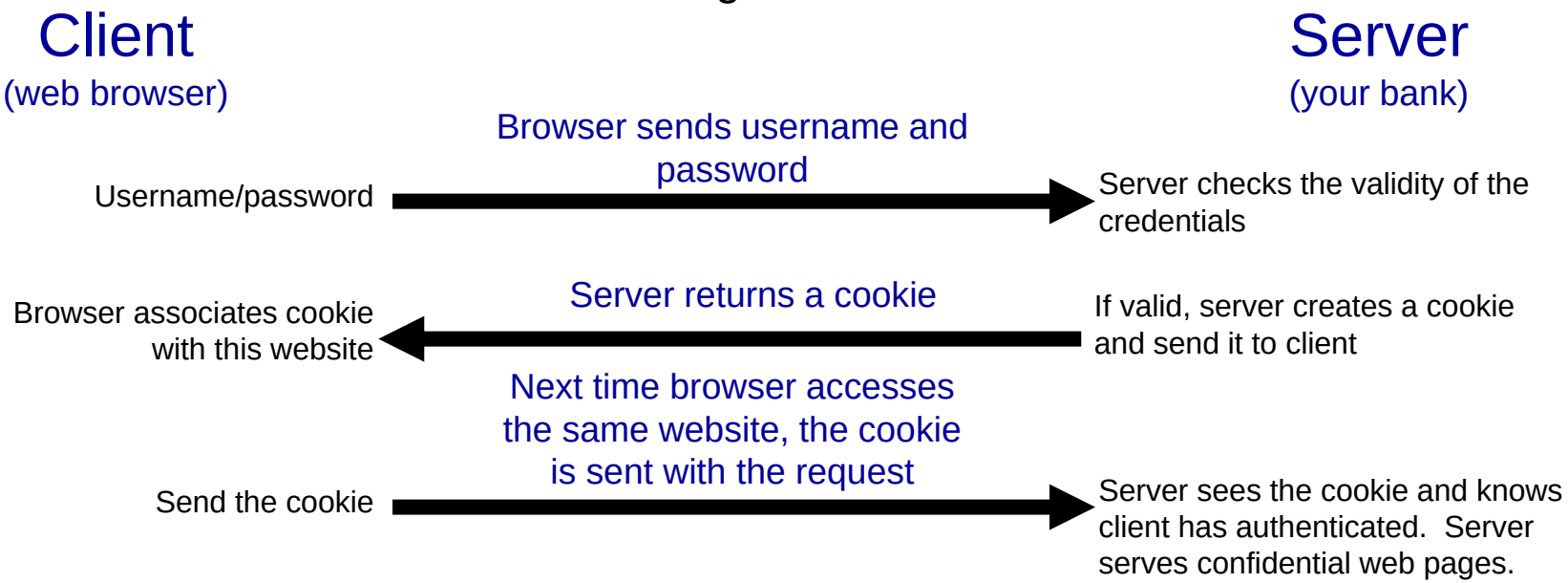
Basic Web Authentication

- Basic web authentication:
  - The first time you visit the website, the browser asks you for your username and password and gives it to the web server
  - But HTTP is stateless, the web server doesn't "remember" that you've logged in when you click on the next link.
  - Your web browser has to "remember" your username and password and send it to the web server every time you access a page from the same website
  - Can be used with SSL so that passwords are transmitted over encrypted channel

- Basic authentication has some serious problems:
  - Every time you access a page, your credentials are transmitted giving an attacker more opportunities to snoop them.
  - Server has no way to end the session without invalidating user's password
  - As a result, basic web authentication is almost NEVER used.

HTTP Cookies

- An HTTP cookie is a piece of data (a big number) that the web server gives the browser. Next time the browser visits the same server, it will send the cookie.
  - This gives the web server a way of tracking which HTTP requests belong to the same user.
  - HTTP cookies can be used for authentication. Once the user has authenticated, the browser gets a cookie.



Advantages and Pitfalls of Using Cookies

- Advantages:
  - **Cookies do not reveal the password**. Even if an adversary can learn the value of the cookie, they do not know the user's username and password.
  - **Cookies can be expired.** If the adversary can learn a valid cookie, the adversary can use it to access private pages. To prevent this, cookies have an expiry time, after which the server will not accept them and the browser will delete them. Too short a time though means the user has to keep entering their password to get a new cookie.
  - **Cookies provide state.** Cookies are really the only practical option to get state in HTTP sessions.

- Like passwords in basic authentication, Cookies for authentication should **always** sent over SSL

Advantages and Pitfalls of Using Cookies

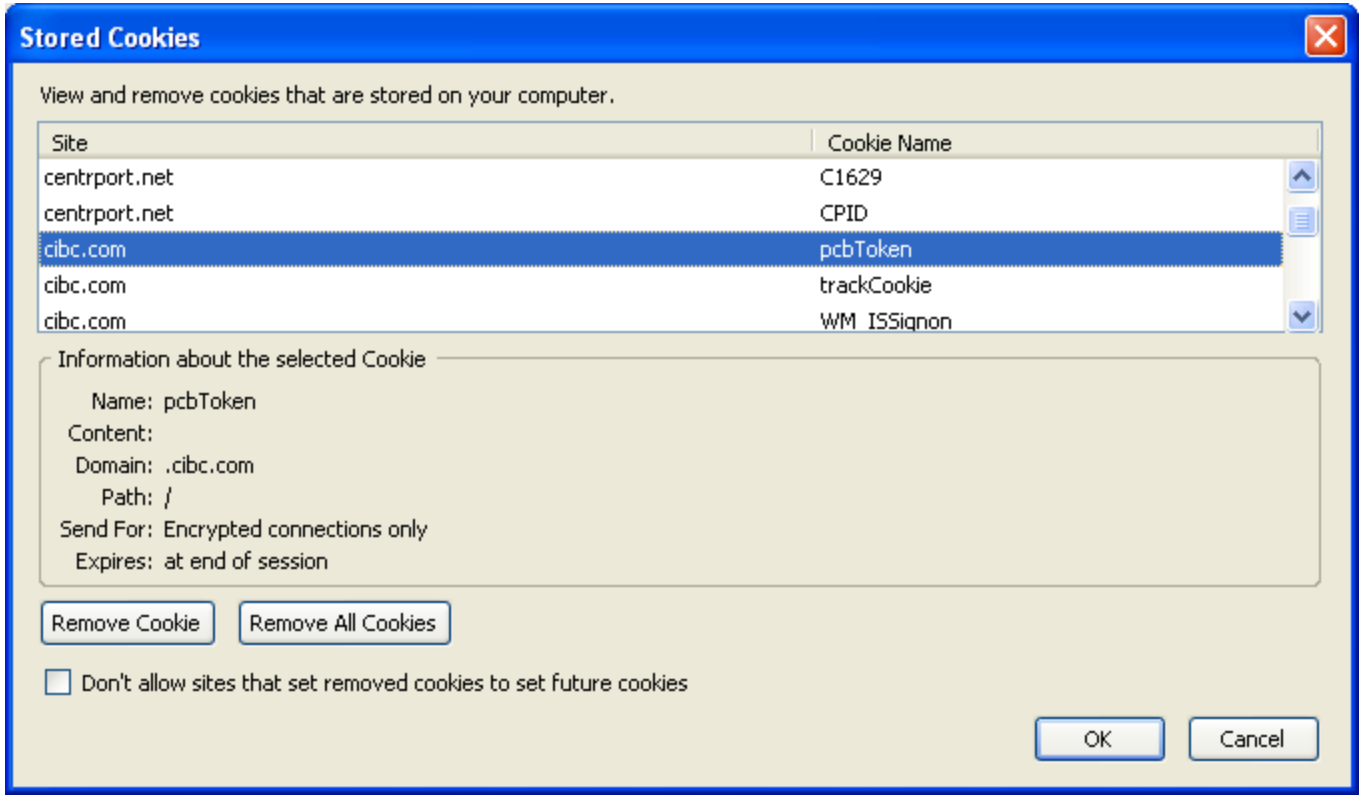
- Pitfalls:
  - **The cookies should not be easy to forge.** Since the web server will accept anyone who presents a valid cookie, it should be hard to "guess" a valid cookie.
  - **Cookies are no good if they aren't sent across SSL.** The web server can specify a policy with the cookie when telling the browser when it should send it. Cookies used in authentication should be specified to be only sent in an encrypted session.
  - **Don't use persistent cookies.** Cookies without an expiration time are persistent. It is not that hard for an attacker to get hold of some cookie files. Making those cookies expire means that the attacker now has to get to the cookie within a certain time period.

Forgeability of Cookies

- Forging a cookie means that an adversary can guess what a valid cookie is without having to get it from the web server.
  - One solution is to make the cookies completely random. However, this means the server has to keep a list what cookies have been issued to which users so that it can identify the user by the cookie presented.
  - Instead, web designers are tempted to include the name of the user in the cookie. However if the cookie just contains the user's name, this makes it too easy for the attacker to forge a cookie (just has to guess a user name)

Cookie creation strategies?

Stored Cookies



Cookies and Privacy

- One of the unrelated drawbacks to cookies is privacy. Web servers can track users as they visit different websites:
  - As part of the policy a web server can specify that a cookie be sent to other web servers. Users can be tracked as they move between web servers.
  - Services like Doubleclick are linked on many different websites. When your browser loads a page with a Doubleclick link, it makes a request from a Doubleclick server and sends any cookies from Doubleclick along with the request. By looking at the URL requested, they can track a single user across websites that use Doubleclick advertising.

Cookie Theft

- A more serious security concern is if an attacker can “steal” a victim’s cookie:
  - If an attacker can somehow get hold of someone’s cookie while it is still valid, the attacker can then access the website using the victim’s identity

- How can an attacker steal a cookie?
  - Cross-site scripting attack
  - Cross-site request forgery

- To understand this, we need to understand a bit about web programming and Javascript