Web Security
XSS – Cross-Site Scripting

• Idea: Place user provided data in the page
• Pros: makes pages more interactive and personal
• Cons: improperly used data could be interpreted as code
• Solutions: Make sure that user data is sanitized and validated
XSSI – Cross-Site Script Inclusion

• Idea: Browsers can prevent pages from one domain to read from pages in another domain
  – Do not prevent pages from referencing resources in other domains (specifically images and scripts)
• Allows an attacker to load their information (via images) or to run their scripts on your site even if you try to block them
• Solution: Make sure the code comes from a trusted site
XSRF – Cross-Site Request Forgery

• Sites will try to protect themselves by only accepting requests that include the proper cookie

• Problem: if the cookie is stolen then an attacker can fake any request and the site will run it

• Solutions: inspect the header, require user-provided secret, add nonce token, etc.
HTTP State

• HTTP is a stateless protocol
  – This means the state machine for the protocol is very simplistic (request and response)
• However, developers want state in order to build staged user experiences
  – Creates a much better user experience.
• Solution: provide state to user and have them echo it back in all future requests.
Option 1: Hidden Fields

• Let's give the user hidden fields that will hold state variables for us to use on later requests

```
<html>
    <head>
        <title>Pay for Pizza</title>
    </head>
    <body>
        <form action="submit_order" method="GET">
            <p> The total cost is 5.50. Are you sure you would like to order? </p>
            <input type="hidden" name="price" value="5.50">
            <input type="submit" name="pay" value="yes">
            <input type="submit" name="pay" value="no">
        </form>
    </body>
</html>
```
Option 1: Problems

Web Browser (Client) ➔ Web Server ➔ Credit Card Payment Gateway

Order 1 Pizza ➔ Confirm $5.50 ➔ Submit Order $5.50

Attacker will modify

Price stored in hidden form variable
Option 1: Problems

Web Browser (Client) → Web Server → Credit Card Payment Gateway

Order 1 Pizza
Confirm $5.50
Submit Order $0.01

Attacker modified price!
Option 2: Cookies

- Stores state on the client side in a special file
- File can only be accessed by code from the same domain
Option 2: Problems

• Cookies can be sniffed from HTTP requests
• Cookies can be stolen from injected scripts
  – LAB 2!
• Not a huge improvement over option 1, except that parameters are not directly visible in Get requests
Option 3: Sessions

- Let’s store state on the server side and only give the user an identifier for it
- Place identifier in a cookie, making it harder to gather
- Make the session id a hash of the user’s IP address and a nonce, making it harder to spoof
Option 3: Problems

• All user state is stored server side
  – This can add up to a lot of data for large sites
• Search for a user’s session data can make the response time very large
• Sessions need to expire, otherwise they could be used by an attacker
• Putting the session id in a cookie does not eliminate XSRF attacks