Exploiting Statelessness

- HTTP is designed to be stateless
- But many useful web interactions are stateful
- Various tricks used to achieve statefulness
  - Usually requiring programmers to provide the state
  - Often trying to minimize work for the server

A Simple Example

- Web sites are set up as graphs of links
- You start at some predefined point
  - A top level page, e.g.
- And you traverse links to get to other pages
- But HTTP doesn't "keep track" of where you've been
  - Each request is simply the name of a link

Why Is That a Problem?

- What if there are unlinked pages on the server?
- Should a user be able to reach those merely by naming them?
- Is that what the site designers intended?

A Concrete Example

- The ApplyYourself system
- Used by colleges to handle student applications
- For example, by Harvard Business School in 2005
- Once all admissions decisions made, results available to students

What Went Wrong?

- Pages representing results were created as decisions were made
- Stored on the web server
  - But not linked to anything, since results not yet released
- Some appliers figured out how to craft URLs to access their pages
  - Finding out early if they were admitted

The Core Problem

- No protocol memory of what came before
- So no protocol way to determine that response matches request
- Could be built into the application that handles requests
- But frequently isn't
  - Or is wrong

Solution Approaches

- Get better programmers
  - Or better programming tools
- Back end system that maintains and compares state
- Front end program that observes requests and responses
  - Producing state as a result
- Cookie-based
  - Store state in cookies (preferably encrypted)

Data Transport Issues

- The web is inherently a network application
- Thus, all issues of network security are relevant
- And all typical network security solutions are applicable
- Where do we see problems?

(Non-) Use of Data Encryption

- Much web traffic is not encrypted
  - Or signed
- As a result, it can be sniffed
- Allowing eavesdropping, MITM attacks, alteration of data in transit, etc.
- Why isn't it encrypted?

Why Web Sites Don't Use Encryption

- Primarily for cost reasons
- Crypto costs cycles
- For high-volume sites, not encrypting messages lets them buy fewer servers
- They are making a cost/benefit analysis decision
- And maybe it's right?

Problems With Not Using Encryption

- Sensitive data can pass in the clear
  - Passwords, credit card numbers, SSNs, etc.
- Attackers can get information from messages to allow injection attacks
- Attackers can readily profile traffic
  - Especially on non-secured wireless networks

Firesheep

- Many wireless networks aren't encrypted
- Many web services don't use end-to-end encryption for entire sessions
- Firesheep was a demo of the dangers of those in combination
- Simple Firefox plug-in to scan unprotected wireless nets for unencrypted cookies
  - Allowing session hijacking attacks
- When run in that environment, tended to be highly successful

Why Does Session Hijacking Work?

- Web sites try to avoid computation costs of encryption
- So they only encrypt login
- Subsequent HTTP messages "authenticated" with a cookie
- Anyone who has the cookie can authenticate
- The cookie is sent in the clear . . .
- So attacker can "become" legit user

Using Encryption on the Web

- Some web sites support use of HTTPS
  - Which permits encryption of data
  - Based on TLS/SSL
- Performs authentication and two-way encryption of traffic
  - Authentication is certificate-based
- HSTS (HTTP Strict Transport Security) requires browsers to use HTTPS

Increased Use of Web Encryption

- These and other problems have led more major web sites to encrypt traffic
- E.g., Google announced in 2014 it would encrypt all search requests
- Facebook, Twitter adopted HSTS in 2014
  - Overall, only 5% of web sites have adopted it as of 2016, though
- Arguably, all web interactions should be encrypted

Sometimes Encryption Isn't Enough

- Especially powerful "attackers" can subvert this process
  - Man-in-the-middle attacks by ISPs
  - NSA compromised key management
  - NSA also spied on supposedly private links
- Usually impossible for typical criminal
- Hard or impossible for a user to know if this is going on

Conclusion

- Web security problems not inherently different than general software security
- But generality, power, ubiquity of the web make them especially important
- Like many other security problems, constrained by legacy issues