### CSci 5271 Introduction to Computer Security Web security, part 2

Stephen McCamant University of Minnesota, Computer Science & Engineering

### Outline

#### Cross-site scripting

More cross-site risks

Announcements intermission

Confidentiality and privacy

Even more web risks

## XSS: HTML/JS injection

Note: CSS is "Cascading Style Sheets"

- Another use of injection template
- Attacker supplies HTML containing JavaScript (or occasionally CSS)
- OWASP's most prevalent weakness in 2017

A category unto itself

Easy to commit in any dynamic page construction

## Why XSS is bad (and named that)

\_\_attacker.com can send you evil JS directly

- 🖲 But XSS allows access to bank.com data
- Violates same-origin policy
- Not all attacks actually involve multiple sites

## Reflected XSS

Injected data used immediately in producing a page
Commonly supplied as query/form parameters
Classic attack is link from evil site to victim site

## Persistent XSS

- Injected data used to produce page later
- For instance, might be stored in database
- Can be used by one site user to attack another user
   E.g., to gain administrator privilege

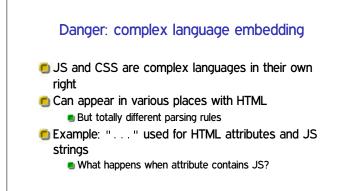
# DOM-based XSS

Injection occurs in client-side page construction

- Flaw at least partially in code running on client
- Many attacks involve mashups and inter-site communication

## No string-free solution

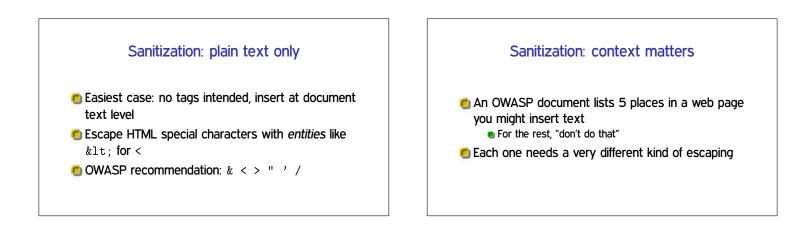
- For server-side XSS, no way to avoid string concatenation
- Web page will be sent as text in the end
   This is the only standard interface
- XSS is an especially hard kind of injection



### Danger: forgiving parsers

History: handwritten HTML, browser competition

- Many syntax mistakes given "likely" interpretations
- Handling of incorrect syntax was not standardized

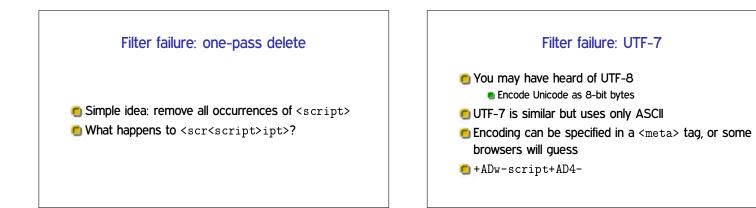


### Sanitization: tag whitelisting

- In some applications, want to allow benign markup like <b>
- But, even benign tags can have JS attributes
- Handling well essentially requires an HTML parser
   But with an adversarial-oriented design

## Don't deny-list

- Browser capabilities continue to evolve
- Attempts to list all bad constructs inevitably incomplete
- Even worse for XSS than other injection attacks



## Filter failure: event handlers

<IMG onmouseover="alert('xss')">

- Put this on something the user will be tempted to click on
- There are more than 100 handlers like this recognized by various browsers

### Use good libraries

Coding your own defenses will never work

- Take advantage of known good implementations
- Best case: already built into your framework Disappointingly rare

### **Content Security Policy**

New HTTP header, W3C candidate recommendation

- Lets site opt-in to stricter treatment of embedded content, such as:
  - No inline JS, only loaded from separate URLs
     Disable JS eval et al.
- Has an interesting violation-reporting mode

#### Outline

Cross-site scripting

- More cross-site risks
- Announcements intermission
- Confidentiality and privacy
- Even more web risks

## HTTP header injection

- Untrusted data included in response headers
- Can include CRLF and new headers, or premature end to headers
- AKA "response splitting"

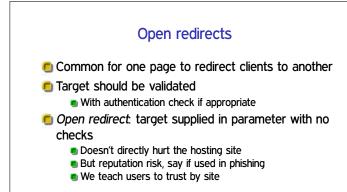
## Content sniffing

- Browsers determine file type from headers, extension, and content-based guessing
   Latter two for ~1% server errors
- Many sites host "untrusted" images and media
- Inconsistencies in guessing lead to a kind of XSS E.g., "chimera" PNG-HTML document

Cross-site request forgery
 Certain web form on bank.com used to wire money
 Link or script on evil.com loads it with certain parameters

 Linking is exception to same-origin
 If I'm logged in, money sent automatically
 Confused deputy, cookies are ambient authority





### Outline

Cross-site scripting

More cross-site risks

Announcements intermission

Confidentiality and privacy

Even more web risks

#### Note to early readers

- This is the section of the slides most likely to change in the final version
- If class has already happened, make sure you have the latest slides for announcements

### Outline

Cross-site scripting

More cross-site risks

Announcements intermission

Confidentiality and privacy

Even more web risks

#### Site perspective

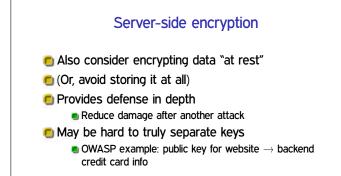
#### Protect confidentiality of authenticators

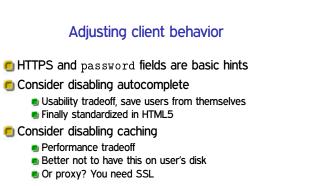
- Passwords, session cookies, CSRF tokens
- Duty to protect some customer info
  - Personally identifying info ("identity theft")
  - Credit-card info (Payment Card Industry Data Security Standards)
  - Health care (HIPAA), education (FERPA)
  - Whatever customers reasonably expect

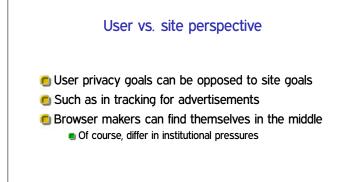
### You need to use SSL

Finally coming around to view that more sites need to support HTTPS

- Special thanks to WiFi, NSA
- If you take credit cards (of course)
- If you ask users to log in
   Must be protecting something, right?
   Also important for users of Tor et al.







## Third party content / web bugs

- Much tracking involves sites other than the one in the URL bar
  - For fun, check where your cookies are coming from
- Various levels of cooperation
- Web bugs are typically 1x1 images used only for tracking

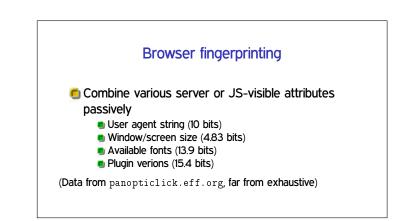
ELike < 0

#### Cookies arms race

- Privacy-sensitive users like to block and/or delete cookies
- Sites have various reasons to retain identification

#### Various workarounds:

- Similar features in Flash and HTML5
- Various channels related to the cache
- $\blacksquare$  Evercookie: store in n places, regenerate if subset are deleted



### History stealing

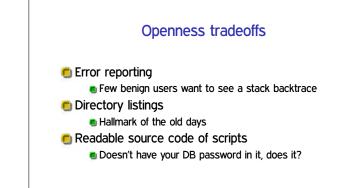
- History of what sites you've visited is not supposed to be JS-visible
- But, many side-channel attacks have been possible
  - Query link color
  - CSS style with external image for visited links
  - Slow-rendering timing channel
  - Harvesting bitmaps
  - User perception (e.g. fake CAPTCHA)

### Browser and extension choices

#### More aggressive privacy behavior lives in extensions

- Disabling most JavaScript (NoScript)
- HTTPS Everywhere (allow-list)
- Tor Browser Bundle
- Default behavior is much more controversial
  - Concern not to kill advertising support as an economic model





## Using vulnerable components

- Large web apps can use a lot of third-party code
- Convenient for attackers too
  - OWASP: two popular vulnerable components downloaded 22m times
- Hiding doesn't work if it's popular
- Stay up to date on security announcements

### Clickjacking

Fool users about what they're clicking on

- Circumvent security confirmations
- Fabricate ad interest

#### Example techniques:

- Frame embedding
- Transparency
- Spoof cursor
- Temporal "bait and switch"

### Crawling and scraping

- A lot of web content is free-of-charge, but proprietary
  - Yours in a certain context, if you view ads, etc.
- Sites don't want it downloaded automatically (web crawling)
- Or parsed and user for another purpose (screen scraping)
- High-rate or honest access detectable