

• • • • • • • •
• • • • • • • •
• • • • • • • •

Cross-Site Scripting Attack (XSS)

• • • • • • • • •
• • • • • • • • •
• • • • • • • • •
• • • • • • • • •
• • • • • • • • •
• • • • • • • • •
• • • • • • • • •
• • • • • • • • •

XSS

❑ Cross-site scripting

- The use of XSS allows spammers to inject executable code on to forums and bulletin boards, which when executed, download and installs malware, steal cookies (sessions), steal hidden data.

XSS

❑ Reflected XSS attack

- Allows executable html script (javascript, VB script) to be injected by the user into a web application.
- When the application replies with a constructed page, the page includes the executable script.
- Useful for tricking users into allowing script-heavy sites to change browser settings.

❑ Stored XSS attack

- Involves executable script being stored on a server (chat room, forum), which executes when it is displayed by another user.

XSS: DOM-based attack

- ❑ Scripts running in your web browser have access to the browser's DOM (document object model), a hierarchy of objects containing everything displayed and stored on each web page in each instance of the browser.
- ❑ Clever scripting can be used to
 - change the contents of the page, adding options, setting default selections.
 - echo/send private data to 3rd parties (similar to stored and reflected attacks).
 - Access the contents of other browser windows/tabs - largely impossible since Google introduced tab sandboxing.

XSS: DOM-based attack

- ❑ DOM-based attacks avoid storage or transmission of javascript. The injected script is not sent to the server, so it is not detected by automated XSS detection.

- ❑ Writable HTML 2 DOM objects:
 - document.location
 - window.location
 - document.url
 - document.urlencoded
 - document.referrer

XSS

❑ Classic example:

- Any html which echoes user-provided text back to the browser without sanitizing it is vulnerable
- `http://www.myforum.org?name=<script>alert(document.cookie);</script>`

❑ Solutions of server side

- Sanitize / validate all input and output.

❑ Solutions of client side

- disable javascript
- noScript plugin for Firefox
- other plugins for other browsers
- Use a serious sanitizing library

<https://github.com/angular/bower-angular-sanitize>

XSS

□ More Solutions:

- Server side: Use **Content Security Policy**
- Set in web server config file
- Apache:

Header set Content-Security-Policy "<*detailed permissions here*>"