Cross-Site Scripting

- XSS
- Many sites allow users to upload information
  - Blogs, photo sharing, Facebook, etc.
  - Which gets permanently stored
  - And displayed
- Attack based on uploading a script
- Other users inadvertently download it
  - And run it . . .

The Effect of XSS

- Arbitrary malicious script executes on user's machine
- In context of his web browser
  - At best, runs with privileges of the site storing the script
  - Often likely to run at full user privileges

Non-Persistent XSS

- Embed a small script in a link pointing to a legitimate web page
- Following the link causes part of it to be echoed back to the user's browser
- Where it gets executed as a script
- Never permanently stored at the server

Persistent XSS

- Upload of data to a web site that stores it permanently
- Generally in a database somewhere
- When other users request the associated web page,
- They get the bad script

Some Examples

- Word Press bug allowed XSS (2016)
- Other XSS vulnerabilities discovered on sites run by Yahoo, Symantec, PayPal, Facebook, LinkedIn, Adobe, Apple App Store, Google Gmail, Fortinet, the Scientology website, thousands of others
- D-Link router flaw exploitable through XSS

Why Is XSS Common?

- Use of scripting languages widespread
  - For legitimate purposes
- Most users leave them enabled in their browsers
- Sites allowing user upload are very popular
- Only a question of getting user to run your script

Typical Effects of XSS Attack

- Most commonly used to steal personal information
  - That is available to legit web site
  - User IDs, passwords, credit card numbers, etc.
- Such information often stored in cookies at client side

Solution Approaches

- Don’t allow uploading of anything
- Don’t allow uploading of scripts
- Provide some form of protection in browser

Disallowing Data Uploading

- Does your web site really need to allow users to upload stuff?
- Even if it does, must you show it to other users?
- If not, just don't take any user input
- **Problem**: Not possible for many important web sites

Don't Allow Script Uploading

- A no-brainer for most sites
  - Few web sites want users to upload scripts, after all
- So validate user input to detect and remove scripts
- **Problem**: Rich forms of data encoding make it hard to detect all scripts
- Good tools can make it easier

Protect the User's Web Browser

- Basically, the same solutions as for any form of protecting from malicious scripts
- With the same problems:
  - Best solutions cripple functionality

Cross-Site Request Forgery

- CSRF
- Works the other way around
- An authenticated and trusted user attacks a web server
  - Usually someone posing as that user
- Generally to fool server into believing that the trusted user made a request

CSRF in Action

- Attacker puts link to (say) a bank on his web page
- Unsuspecting user clicks on the link
- His authentication cookie goes with the HTTP request
  - Since it's for the proper domain
- Bank authenticates him and transfers his funds to the attacker

Issues for CSRF Attacks

- Not always possible or easy
- Attacks sites that don't check referrer header
  - Indicating that request came from another web page
- Attacked site must allow use of web page to allow something useful (e.g., bank withdrawal)
- Must not require secrets from user
- Victim must click link on attacker's web site
- And attacker doesn't see responses

CSRF In the Wild

- CSRF possibility in Verizon Mobile App API
- eBay CSRF problem in Magneto e-commerce system
- A CSRF-based pharming toolkit that attacked wireless routers discovered
- CSRF problem in Arris cable modems