A8: Cross-site Request Forgery (CSRF)

A8: Cross-site Request Forgery (CSRF)

- XSS
  - Trick browser to execute code without user knowledge
- CSRF
  - Trick browser to access sensitive pages without user knowledge

CSRF Vulnerability Pattern

- Problem
  - Web browsers automatically include most credentials with each request
    - Session cookie
    - Basic authentication header
  - Even for requests caused by a form, script, or image from another site
- Sites relying solely on automatic credentials are vulnerable!

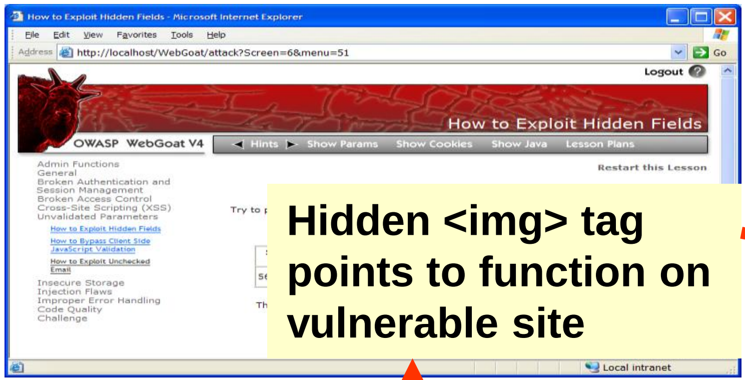
CSRF Illustrated

1

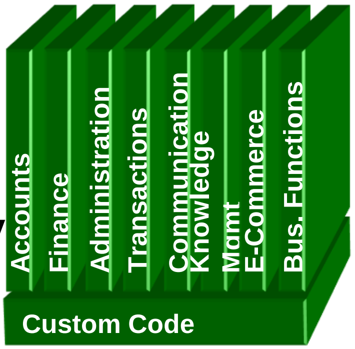
Attacker finds function on vulnerable site he wants victim to hit while authenticated

Sets a trap via a website or e-mail

Site with CSRF vulnerability


How to Exploit Hidden Fields - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/WebGoat/attack?Screen=6&menu=51
Go
Logout
How to Exploit Hidden Fields
OWASP WebGoat V4
Hints
Show Params
Show Cookies
Show Java
Lesson Plans
Restart this Lesson
Admin Functions
General
Broken Authentication and Session Management
Broken Access Control
Cross-Site Scripting (XSS)
Unvalidated Parameters
How to Exploit Hidden Fields
How to Bypass Client Side JavaScript Validation
How to Exploit Unchecked Email
Insecure Storage
Injection Flaws
Improper Error Handling
Code Quality
Challenge
Local intranet

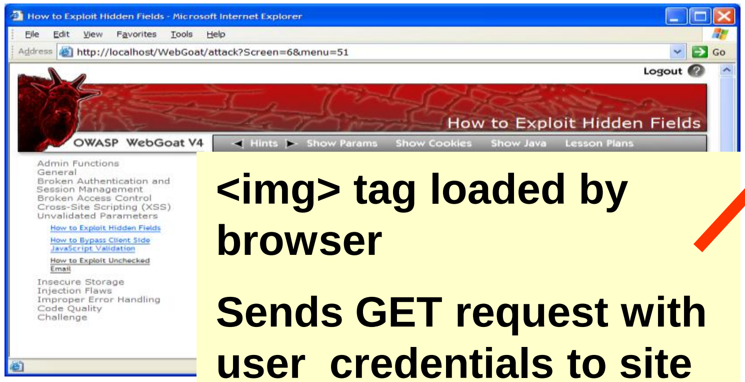
Hidden <img> tag points to function on vulnerable site



2

While logged into site with CSRF vulnerability

Victim views attacker site


How to Exploit Hidden Fields - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/WebGoat/attack?Screen=6&menu=51
Go
Logout
How to Exploit Hidden Fields
OWASP WebGoat V4
Hints
Show Params
Show Cookies
Show Java
Lesson Plans
Admin Functions
General
Broken Authentication and Session Management
Broken Access Control
Cross-Site Scripting (XSS)
Unvalidated Parameters
How to Exploit Hidden Fields
How to Bypass Client Side JavaScript Validation
How to Exploit Unchecked Email
Insecure Storage
Injection Flaws
Improper Error Handling
Code Quality
Challenge

<img> tag loaded by browser

Sends GET request with user credentials to site

3

Vulnerable site sees legitimate request from victim and performs the action requested

Example

- Trick user with account at `bank.cxx` to visit your rogue page

```
<html><body>
<img
src=https://www.bank.cxx/transfer_funds?amount=1000&to_account=12345678 />
</body></html>
```

- If user previously logged into `www.bank.cxx`, transfer occurs unbeknownst to user

Common CSRF activities

- **Initiate transactions (transfer funds, logout user, close account)**
- **Access sensitive data**
- **Change account details**

A8 - Prevention

- http://www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet

Secret tokens

- Add a secret token to origin page of ALL sensitive requests
  - Attacker can't spoof the request unless there's an XSS hole in origin page of client that leaks secret.
  - Tokens should be cryptographically secure (random hash or number)
- Examples
  - Add secret token into all forms and links
  - Like setting a cookie on client page itself
    - Hidden Field

```
<input name="token" value="687965fdfaew87agrde" type="hidden"/>
```

  - Ensure token never exposed via referer header or in the clear
    - Example: Should not appear in a GET-based form submission:

```
/accounts?token=687965fdfaew87agrde …
```

  - Have a unique token for each function
    - Use a hash of function name, session id, and a secret to generate
- Attacker unable to get victim to send validating secret token

Server methods

- Only use HTTP GET for “safe methods”
  - Methods that have no persistent side effects on server
  - Rely upon HTTP POST requests with tokens for actions with persistent side-effects
- Require secondary authentication for sensitive functions (e.g., eTrade)
- Expire authorization cookie quickly if session is idle

Homework

- See handout

Questions

- https://sayat.me/wu4f