Cross Site Request Forgery
HTML Forms

• Allow a user to provide some data which gets sent with an HTTP POST request to a server

<form action="bank.com/action.php">

First name: <input type="text" name="firstname">

Last name: <input type="text" name="lastname">

<input type="submit" value="Submit"></form>

When filling in Alice and Smith, and clicking submit, the browser issues

HTTP POST request

bank.com/action.php?firstname=Alice&lastname=Smith

As always, the browser attaches relevant cookies
Consider the cookie stores the session token

- Server assigns a random session token to each user after they logged in, places it in the cookie
- The server keeps a table of [username -> session token], so when it sees the session token it knows which user
- When the user logs out, the server clears the session token
Session using cookies

Browser → Server

POST/login.cgi

Set-cookie: session token

Server → Browser

GET/POST...
Cookie: session token

response
CSRF Attack Basic Picture

1. establish session
2. visit server
3. receive malicious page (cookie for bank.com with session token)
4. send forged request (w/ cookie)

What can go bad? URL contains transaction action
Cross Site Request Forgery (CSRF)

- User logs in to bank.com
  - Session cookie remains in browser state

- User visits malicious site containing:
  <form name=F action=http://bank.com/BillPay.php>
    <input name=recipient value=badguy> …
    <script> document.F.submit(); </script>

- Browser sends user auth cookie with request
  - Transaction will be fulfilled

- Problem:
  - cookie auth is insufficient when side effects occur
Form post with cookie

www.attacker.com

Victim Browser

GET /blog HTTP/1.1

www.bank.com
Form post with cookie

www.attacker.com

GET /blog HTTP/1.1

User credentials

www.bank.com

POST /transfer HTTP/1.1
Referer: http://www.attacker.com/blog
recipient=attacker&amount=$100
Cookie: SessionID=523FA4cd2E

Transfer complete!
An attacker could
• add videos to a user’s "Favorites,"
• add himself to a user’s "Friend" or "Family" list,
• send arbitrary messages on the user’s behalf,
• flagged videos as inappropriate,
• automatically shared a video with a user’s contacts,
  subscribed a user to a "channel" (a set of videos published by one person or group), and
• added videos to a user’s "QuickList" (a list of videos a user intends to watch at a later point).
Facebook Hit by Cross-Site Request Forgery Attack

By Sean Michael Kerner  |  August 20, 2009

September 30, 2008

Popular websites fall victim to CSRF exploits
CSRF Defenses

• CSRF token

• Referer Validation

• Others (e.g., custom HTTP Header) we won’t go into
1. goodsite.com server wants to protect itself from CSRF attacks, so it includes a secret token into the webpage (e.g., in forms as a hidden field)

2. Requests to goodsite.com include the secret

3. goodsite.com server checks that the token embedded in the webpage is the expected one; reject request if not.

Can the token be?

- 123456
- Dateofbirth

CSRF token must be hard to guess by the attacker
How the token is used

- The server stores state that binds the user's CSRF token to the user's session token
- Embeds a fresh CSRF token in every form
- On every request the server validates that the supplied CSRF token is associated with the user's session token
- Disadvantage is that the server needs to maintain a large state table to validate the tokens.
Cookie: SessionID = 523FA4cd2E

Regular use

GET page

page with form
hidden field
CSRF 198..

Victim Browser

www.bank.com

cookie for bank.com: SessionID = 523FA4cd2E

Recipient = Alice & amount = 10 & CSRF = 198..

Session ID
523..

CSRF token
198..
Cookie: SessionID=523FA4cd2E

Attack attempt

www.attacker.com

GET /blog HTTP/1.1

Victim Browser

www.bank.com

Session ID | CSRF token
--------- | --------
523.. | 198..

doesn't know CSRF token for victim's session (one might not even be set at the server if the user did not request the form recently)
Other CRSF protection: Referer Validation

- When the browser issues an HTTP request, it includes a referer header that indicates which URL initiated the request.
- This information in the Referer header could be used to distinguish between same site request and cross site request.
Refer header

```
<form action=https://www.bank.com/transfer method=POST target=invisibleframe>
  <input name=recipient value=attacker>
  <input name=amount value=$100>
</form>
<script>document.forms[0].submit()</script>
```

POST /transfer HTTP/1.1
Referer: http://www.attacker.com/blog
Cookie: SessionID=523FA4cd2E

GET /blog HTTP/1.1
HTTP/1.1 200 OK
Transfer complete!
Referer Validation

Facebook Login

For your security, never enter your Facebook password on sites not located on Facebook.com.

Email: [input field]
Password: [input field]

Remember me
Login or Sign up for Facebook

Forgot your password?
Referer Validation Defense

• HTTP Referer header
  – Referer: http://www.facebook.com/ ✔
  – Referer: [empty]
    • Strict policy disallows (secure, less usable)
    • Lenient policy allows (less secure, more usable)
Privacy Issues with Referer header

- The referer contains sensitive information that impinges on the privacy
- The referer header reveals contents of the search query that lead to visit a website.
- Some organizations are concerned that confidential information about their corporate intranet might leak to external websites via Referer header
Referer Privacy Problems

• Referer may leak privacy-sensitive information
  

• Common sources of blocking:
  – Network stripping by the organization
  – Network stripping by local machine
  – Stripped by browser for HTTPS -> HTTP transitions
  – User preference in browser
Summary: CSRF

- CSRF attacks execute request on benign site because cookie is sent automatically.
- Defenses for CSRF:
  - embed unpredictable token and check it later.
  - check referer header in addition as defense in depth.