# Defending against Session fixation and hijacking

<https://ferdous.wordpress.com/2008/12/12/session-fixation-session-hijacking/>

<http://www.dreamincode.net/forums/topic/61503-guarding-against-session-hijacking-in-aspnet/>

<http://www.codeproject.com/Articles/210993/Session-Fixation-vulnerability-in-ASP-NET>

http://lockmedown.com/preventing-xss-in-asp-net-made-easy/

Don’t put session IDs in the URL. Django explicitly does not support this because it’s just dangerous.

Use SSL and secure cookies.

Use HttpOnly cookies.

Use secure cookies

Session ID Name Fingerprinting( Name should be generic, it should not disclose any technologies or languages used)

Session Id length should be at least 128 bits(16 bytes)

Renew the Session ID After Any Privilege Level Change

# Defending against Clickjacking

There are two main ways to prevent clickjacking:

Sending the proper X-Frame-Options HTTP response headers that instruct the browser to not allow framing from other domains

Employing defensive code in the UI to ensure that the current frame is the most top level window

# Defending against XSS

Sanitize user inputs(whiltlisting)

Implement correct output encoding

html escaping, javascript escaping, attribute escape

AntiXssEncoder to encode strings before saving into database

http://stackoverflow.com/questions/24502609/how-can-i-sanitize-input-on-a-webapi-model

# Defending against CSRF

Anti forgery token

Host and referrer check(both should be pointing to same domain)

<http://stackoverflow.com/questions/19788916/how-to-make-ajax-request-with-anti-forgery-token-in-mvc>

Challenge- Response method:

Re-authentication

CAPTCHA

# Defending against insecure direct object reference:

Implement proper access control

# Defending against Sql injection:

Parameterization of sql queries

LINQ to SQL passes all data to the database via SQL parameters. So, although the SQL query is composed dynamically, the values are substitued server side through

parameters safeguarding against the most common cause of SQL injection attacks.

# Defend against broken authentication and session management:

Cross-Origin resource sharing:

Any inappropriate domains should be removed from the CORS policy.

Security measures to be taken for SSL:

SSL:Reject invalid certificates ( if proxy request through fiddler then fiddler generates a new certificate )

Certificate pinning

Never disable certificate validation

Cookie Injection Attack:

Use httponly

Trust boundary violation:

Define clear trust boundaries in the application. Do not use the same data structure to hold trusted data in some contexts and untrusted data in other contexts. Minimize the number of ways that data can move across a trust boundary.