* Sql Injection

Säkra sidan använder vi Linq-queries. Osäkra sidan, använder vi Sql Statements.

* XSS

Säkra sidan gör vi inga förändringar på, den är per default säker mot XSS. Osäkra sidan skriver vi ut en variabel med hjälp av @HTML.Raw, och stänger av den automatiska valideringen med hjälp av attributet ValdidateInput = false, på den berörda actionmetoden.

XSS attackerna kommer kunna utföras på ProductInfo vyn genom att skriva en Utvärdering (Review).

* CSRF

På den säkra sidan kommer vi att användas oss av Html.AntiForgeryToken som kommer att läggas i vår vy där du har möjligheten att ändra värden. Vi kommer även att lägga ValidateAntiForgeryTokenAttribute till vår post-action som är kopplad till den vy där du kan ändra värden. På den osäkra sidan kommer vi helt enkelt inte att ha något att dessa attribut/metoder.

* Missing Function Level Accsess Control

Vi tänker oss att vi har en viss funktion som bara en administratör ska kunna utföra, men på den osäkra sidan kan man med viss manipulering ändra URL-adressen så att man kan även om man inte har behörighet utföra den sagda funktionen. På den säkra sidan kommer vi att utföra en kontroll om huruvida man har behörighet eller ej att utföra funktionen.

* Insecure DIrect Object Reference

Med detta menas att man kan komma åt filer eller dokument på sidan pga otillräckliga filter i i webconfig-filen alternativt appconfig-filen. På den säkra sidan har vi per default ett filter som skyddar känsliga filer, såsom web.config, bin, app.config osv.

* Egendefinierad felsida

På den säkra sidan så hänvisas du till en egendefinierad felsida, när ett fel uppstår. På den osäkra sidan, implementerar vi inte någon egendefinierad felsida. Därigenom kan du bland annat utläsa vilket ramverk vi använder.