

Capítulo 5: Aplicando técnicas en una aplicación ASP.NET MVC

Capítulo 6: Seguridad de Aplicaciones Web

Capítulo 7: Programación del lado del Cliente

# Seguridad de Aplicaciones Web

## Visual Studio 2017 Web Developer



# Objetivos

Al finalizar el capítulo, el alumno:

- Implementa autenticación y autorización.
- Usa ASP.NET Identity en una aplicación MVC.
- Trabaja con Áreas.
- Evita ataques a las aplicaciones



# Agenda

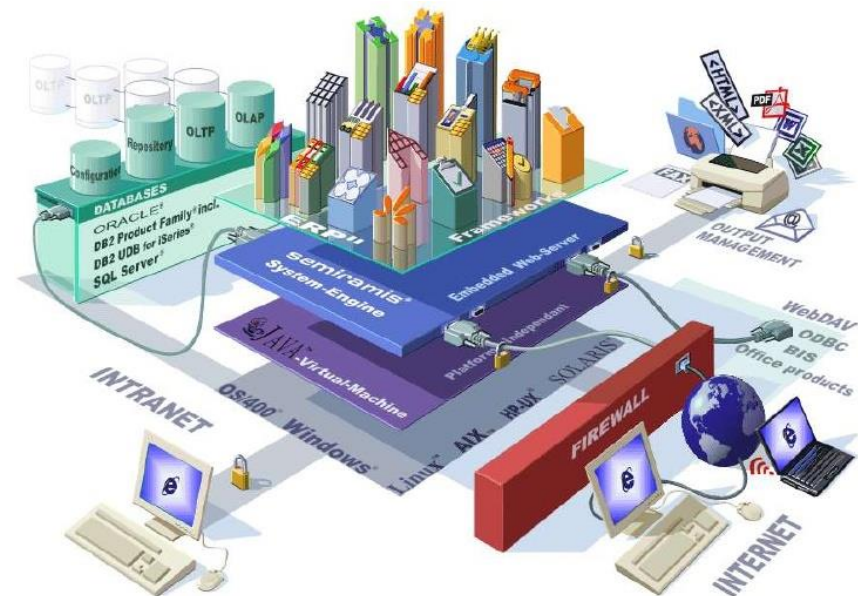
- Introducción a la Seguridad en Aplicaciones
- Autenticación y Autorización
- ASP.NET Identity
- Áreas
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)



# Introducción a la Seguridad en Aplicaciones

La creación de una aplicación web segura no solo implica crear usuarios e implementar una página de login.

Es necesario familiarizarse con temas de seguridad que ofrece el sistema operativo, en este caso, Windows.



# Introducción a la Seguridad en Aplicaciones

## Criterios a tener en cuenta para proteger una aplicación web

- Tener Windows, IIS, SQL Server, etc. actualizados.
- Ejecutar las aplicaciones con los privilegios mínimos necesarios (por ejemplo, nuestra aplicación web no debería conectarse con el usuario administrador a la base de datos).
- Conocer a los usuarios y manejar perfiles de privilegios.
- Tener acceso seguro a bases de datos.
- Crear mensajes de error adecuados, sin dar detalles de la implementación interna de la aplicación (por ejemplo, no se debe mostrar el stack trace de las excepciones).
- Proteger la información confidencial.
- Usar cookies de forma segura.
- Implementar técnicas para protegerse contra ataques o amenazas.



# Introducción a la Seguridad en Aplicaciones

## Clasificación de amenazas

- Suplantación.  
Manipulación.
- Repudio.
- Revelación de información.
- Denegación de servicio.
- Concesión de privilegio.



# Autenticación y Autorización

En la seguridad de las aplicaciones de cualquier tipo y cualquier tecnología siempre se debe tener en cuenta dos conceptos claves, **autenticación** y **autorización**.

## Autenticación

Es el proceso de identificar quién está accediendo a la aplicación. En Visual Studio 2017 se permite elegir entre:

- Individual User Accounts.
- Work And School Accounts.
- Windows Authentication



# Autenticación y Autorización

## Autorización

Es el proceso de determinar qué acciones puede realizar un usuario autenticado, de acuerdo a los permisos que tiene asignado.





# Autenticación y Autorización

## ASP.NET Identity

Para manejar de manera más rápida y fácil el aspecto de Autenticación y Autorización en aplicaciones Web, ASP.NET incluía ASP.NET Membership (hasta ASP.NET 3.5), esto incluía un panel de administración de permisos en el cual registrábamos usuarios, roles, esto se registraba en un esquema de datos ya definido. Posteriormente, se desarrolló ASP.NET Simple Membership (ASP.NET 4) y ahora se tiene **ASP.NET Identity** (ASP.NET 4.5 y 5).

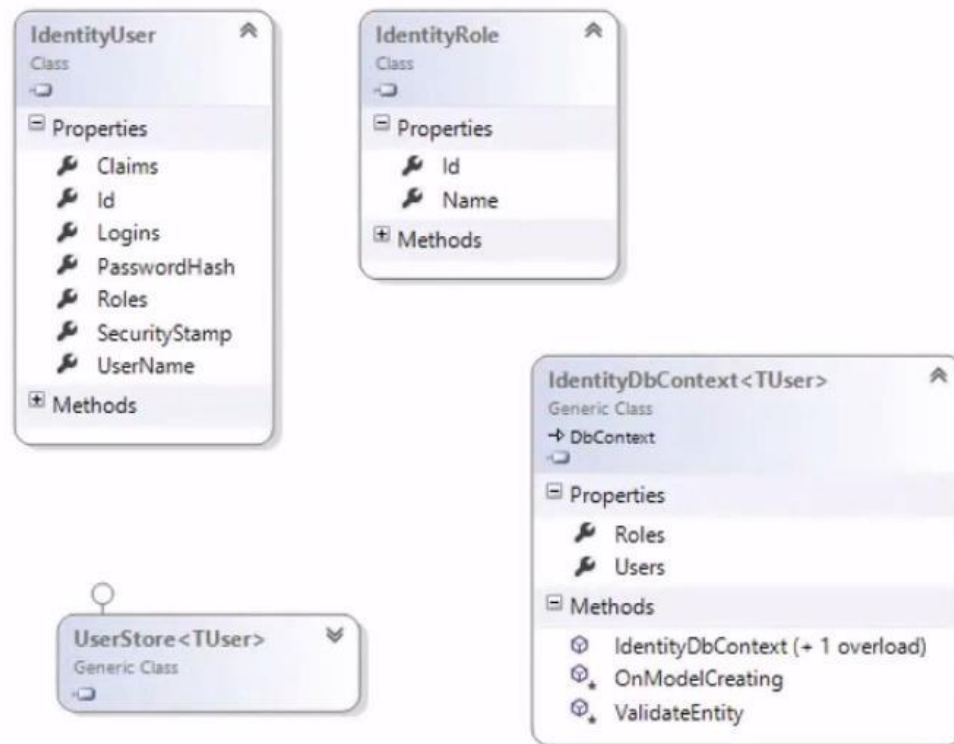
ASP.NET Identity también incluye un esquema de base de datos predefinido que podemos utilizar inmediatamente con la configuración predeterminada de nuestros proyectos. Utiliza **Entity Framework Code First** para crear la base datos.



# Autenticación y Autorización

## ASP.NET Identity

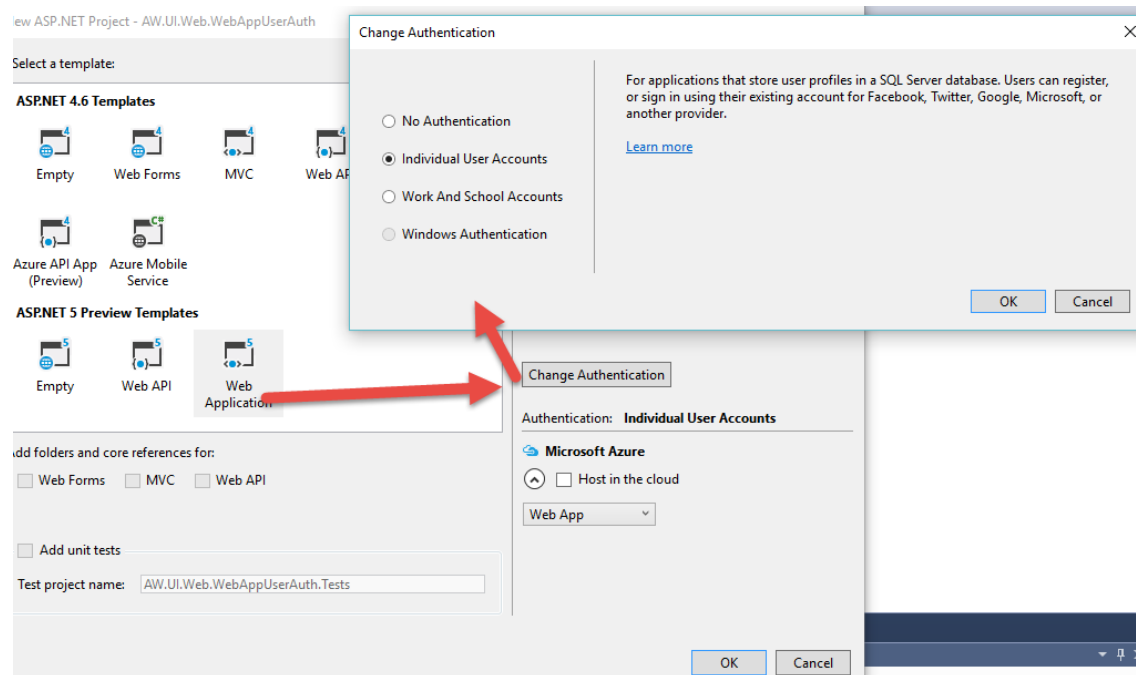
Microsoft.AspNet.Identity.EntityFramework



# Autenticación y Autorización

## ASP.NET Identity

Para aplicaciones que se usen en internet, debemos implementar la autenticación por formularios (Individual User Accounts)



# Autenticación y Autorización

## ASP.NET Identity

Archivos autogenerados:

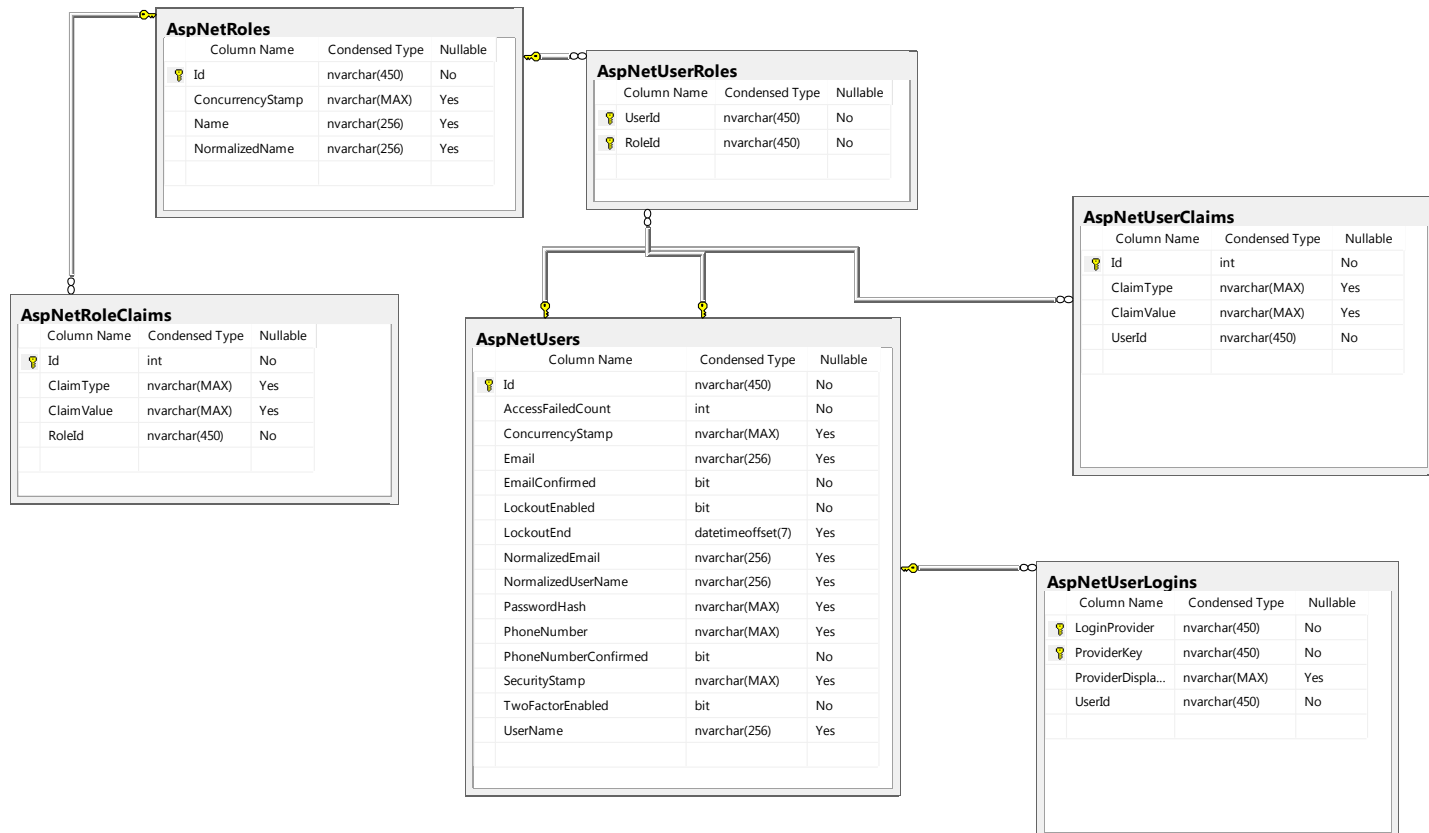
- AccountController: Contiene los action methods Login, Logoff, Register, entre otros.
- AccountViewModels: Contiene los ViewModels a ser usados por las vistas generadas.
- IdentityModels: Contiene el IdentityDbContext (DbContext de entity framework especializado para ASP.NET Identity).
- Account Views: Contiene las vistas que se generan automáticamente, y ya traen la funcionalidad lista para usar.



# Autenticación y Autorización

## ASP.NET Identity

La base de datos que se genera tiene las siguientes tablas:



# Autenticación y Autorización

## ASP.NET Identity

Autorización en Asp.Net Identity:

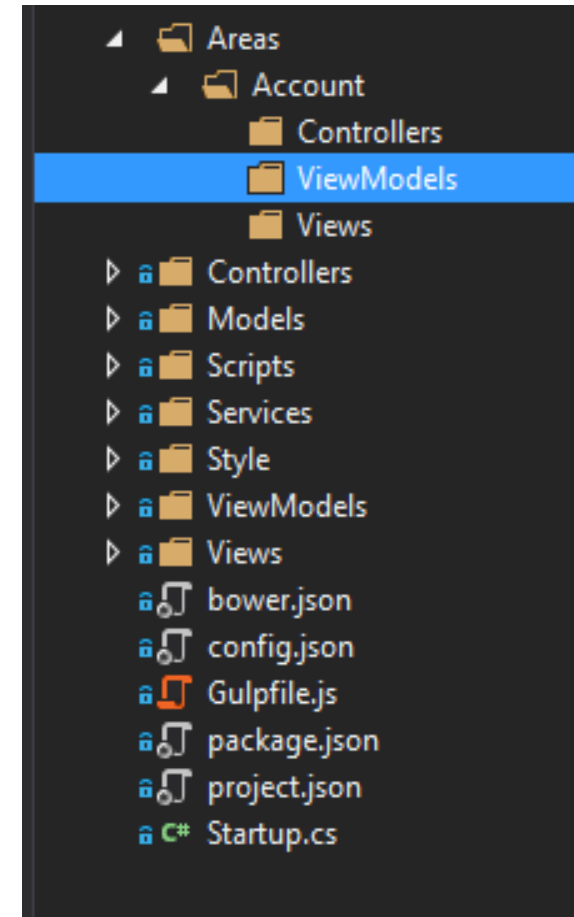
```
public class HomeController : Controller
{
    [Authorize(Roles="Administrators")]
    0 references
    public ActionResult Index()
    {
```

```
[Authorize]
0 references
public class HomeController : Controller
{
    [AllowAnonymous]
    0 references
    public ActionResult Index()
    {
```



# Áreas

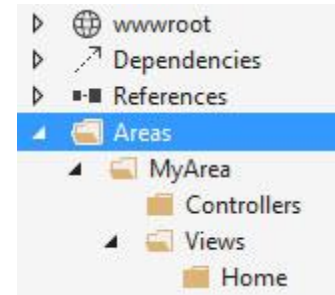
- ASP.NET MVC le permite dividir las aplicaciones Web en unidades más pequeñas que se conocen como áreas.
- Las Áreas proporcionan una manera de separar una aplicación Web grande MVC en agrupaciones funcionales más pequeñas.
- Un área es una estructura MVC dentro de una aplicación. Una aplicación puede contener varias estructuras MVC (áreas).



# Áreas

Pasos para trabajar con áreas:

1. Crear un folder en la carpeta raíz, con el nombre **Áreas** y dentro de esta las áreas que desea implementar, como la imagen tenemos Areas->MyArea.
2. Los controladores a crear deben agregarse en la carpeta Controllers del área a trabajar. La diferencia con un controlador normal es que ahora estos controladores deben tener el atributo [Area].



```
[Area("MyArea")]
public class HomeController : Controller
{
    // GET: /<controller>/
    public IActionResult Index()
    {
        return View();
    }
}
```





# Áreas

3. Como último paso se tiene que configurar la ruta del área en el método Configure del archivo Startup.cs.

```
// Add MVC to the request pipeline.
app.UseMvc(routes =>
{
    // add the new route here.
    routes.MapRoute(name: "areaRoute",
        template: "{area:exists}/{controller}/{action}",
        defaults: new { controller = "Home", action = "Index" });

    routes.MapRoute(
        name: "default",
        template: "{controller}/{action}/{id?}",
        defaults: new { controller = "Home", action = "Index" });
});
```

Ahora la nueva ruta debe trabajar exactamente como las rutas por defecto pero ahora se le tiene que adicionar el nombre del área. Así, usted puede crear una vista Index para su controlador HomeController y navegar de la siguiente manera /MyArea/Home or /MyArea/Home/Index.



# Cross-Site Scripting (XSS)

Uno de los aspectos que se deben controlar en el desarrollo de una aplicación es la validación de entrada de datos, y el Cross-Site Scripting (XSS) es una técnica utilizada para inyectar código malicioso ejecutable en las aplicaciones, del cual todo desarrollador se debe proteger.

```
public class Post
{
    [Key]
    public int Id { get; set; }

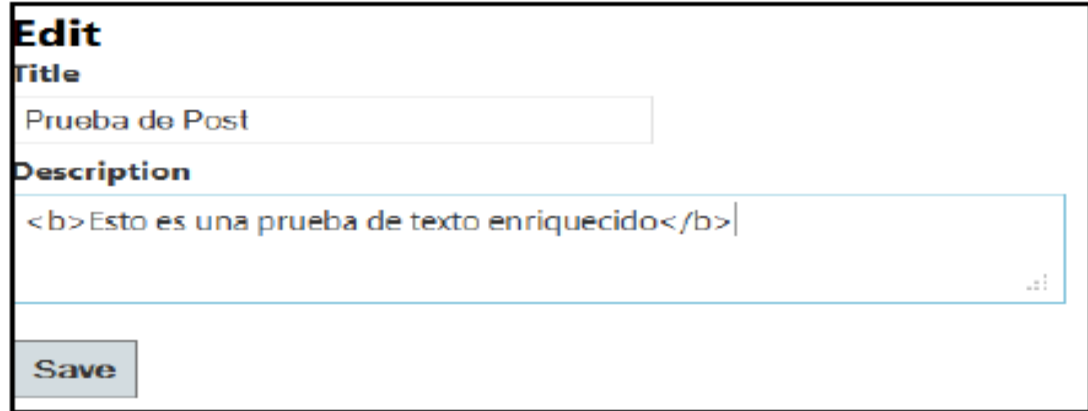
    [Required]
    public string Title { get; set; }

    [DataType(DataType.MultilineText)]
    [AllowHtml]
    public string Description { get; set; }
}
```



# Cross-Site Scripting (XSS)

Código malicioso:



**Edit**

**Title**


Prueba de Post

**Description**

<b>Esto es una prueba de texto enriquecido</b>

Save

Se obtiene un error:



Error de servidor en la aplicación '/'.

Se detectó un posible valor Request.Form peligroso en el cliente (Description= "<b>Esto es una prueba de texto enriquecido</b>")

**Descripción:** La validación de solicitudes ha detectado un valor de entrada del cliente potencialmente peligroso, y se ha anulado el procesamiento de la solicitud. Este valor puede invalidar la configuración de validación de solicitudes de la aplicación, establezca el atributo requestValidationMode de la sección de configuración httpRuntime en requestWebResource en el archivo WebResource.config para deshabilitar la validación de solicitudes estableciendo validateRequest="false" en la directiva Page o en la sección de configuración <pages>. Sin embargo, es absolutamente recomendable que se informe de esta información, vea <http://go.microsoft.com/fwlink?LinkID=153133>.

**Detalles de la excepción:** System.Web.HttpRequestValidationException: Se detectó un posible valor Request.Form peligroso en el cliente (Description= "<b>Esto es una prueba de texto enriquecido</b>")

**Error de código fuente:**

Se ha generado una excepción no controlada durante la ejecución de la solicitud Web actual. La información sobre el origen y la ubicación de la excepción pueden identificarse en el archivo de eventos de depuración.

**Seguimiento de la pila:**

[HttpRequestValidationException (0x80004005): Se detectó un posible valor Request.Form peligroso en el cliente (Description= "<b>Esto es una prueba de texto enriquecido</b>")]  
System.Web.HttpRequest.ValidateString(String value, String collectionKey, RequestValidationSource request)



# Cross-Site Scripting (XSS)

ASP.NET MVC protege por defecto del XSS. Sin embargo, el problema se presenta si se desea que los usuarios puedan introducir código HTML en la aplicación.

Para que los usuarios puedan insertar texto enriquecido, como por ejemplo HTML, se deberá realizar lo siguiente:

- Decorar las propiedades de las clases del modelo con [AllowHtml].
- Utilizar en las vistas el Helper @Html.Raw(Propiedad).



# Cross-Site Scripting (XSS)

- **AntiXSS** que provee Microsoft y que se encuentra en NuGet, la cual provee de todos los elementos necesarios para proteger contra XSS.
- AntiXSS añade dos referencias al proyecto: AntiXSSLibrary y HtmlSanitizationLibrary, que se utilizan en los controllers para sanear los datos de entrada que puedan contener código malicioso.

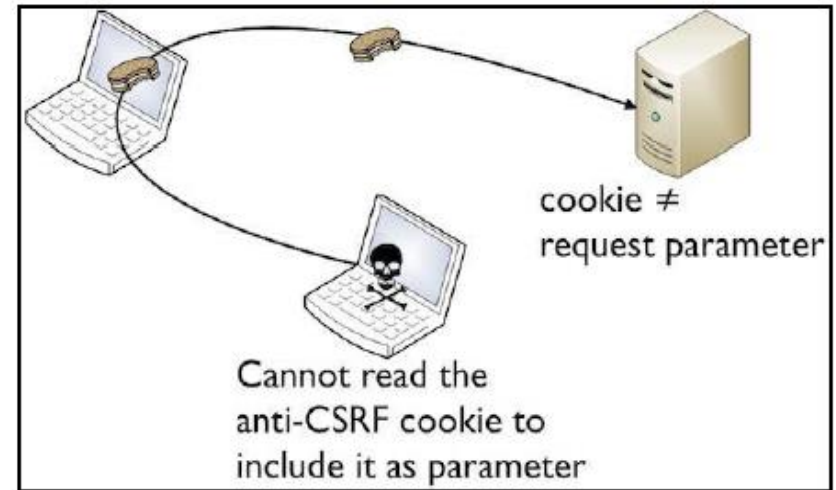
```
public ActionResult Create([Bind(Include="Id,Title,Description")] Post post)
{
    if (ModelState.IsValid)
    {
        db.Posts.Add(post);
        post.Description = Sanitizer.GetSafeHtmlFragment(post.Description);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(post);
}
```



# Cross-Site Request Forgery (CSRF)

- El Cross-Site Request Forgery (CSRF o XSRF), al contrario del XSS que explota la confianza del usuario en un sitio en particular, explota la confianza del sitio en un usuario en particular.
- **El CSRF aprovecha que un usuario está validado en una aplicación**, para a través de esta, introducir solicitudes "válidas" que modifiquen el comportamiento de la aplicación a favor del atacante. Es decir, el atacante usa a la víctima para que sea ella misma quien realice la transacción dañina, cuando la víctima se encuentra validada en el servidor y en la aplicación específica.



# Cross-Site Request Forgery (CSRF)

En el caso de ASP.NET MVC 6, se debe agregar el atributo `ValidateAntiForgeryToken` en la acción del controlador a validar CSRF, tal como se muestra en el siguiente ejemplo:

```
public class MyController : Controller
{
    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Create(SomeModel model)
    {
        ...
    }
}
```



# Cross-Site Request Forgery (CSRF)

- El resultado del renderizado del html sería el siguiente:

```
<form action="/My/Create/" method="post">

<input name="__RequestVerificationToken" type="hidden"
value="CfDJ8DCDWZ4iOzZDmNK15HFAUd2qQe4qwOhVP4znwD1TDINNk_h-
1a2v0A1aDPCdb41Egc9X_cTsJkKCDUCYh9EKr7HqXI3hvIRfnRItwfJwbImCZIx38uDf
wVu5jzdVZOcaXUUmoPBDtG6-0__0FVezb-U" />

</form>
```

- Si se desea deshabilitar el token de seguridad, se tiene que indicar una propiedad asp-antiforgery en el TagHelper del form.

```
<form asp-action="Create" asp-antiforgery="false">

</form>
```





# OWASP ZAP

Una herramienta de penetración para ubicar vulnerabilidades en aplicaciones web .

- Su prioridad es la facilidad de uso.
- Paginas de ayuda simples.
- Gratis
- Open source
- Multiplataforma
- En constante desarrollo
- Mucha participación Activa



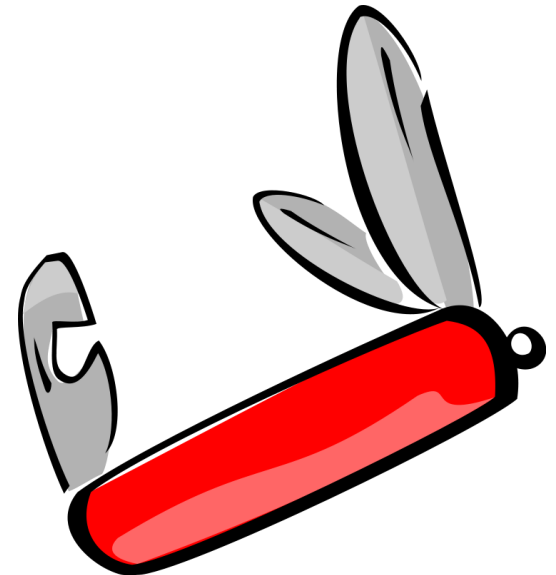
# Características

Todo lo necesario para probar una aplicacion web.

- Intercepting proxy
- Active scanner
- Passive scanner
- Spider
- Brute force (OWASP DirBuster code)
- Port Scanner

Y muchas heraamientas adicionales:

- Auto tagging
- Report generation
- Session comparison
- Smart card support



# **Ejercicio N° 6.1: Implementación de autenticación y creación la página de Login**

Haciendo uso de ASP.NET Identity implementa autenticación y autorización en nuestro proyecto actual.

Al finalizar el laboratorio, el alumno logrará:

- Conocer los conceptos de seguridad de aplicaciones.
- Conocer cómo trabajar ASP.NET Identity en aplicaciones web.



## **Ejercicio N° 6.2: Instalar y validar reporte de OWASP Zed Attack Proxy**

Realizará un scanner de nuestra aplicación actual para ubicar vulnerabilidades.

Al finalizar el laboratorio, el alumno logrará:

- Instalar y usar OWASP Zed Attack Proxy para poder identificar posibles vulnerabilidades en aplicaciones web.



## **Tarea N° 6.1: Crea un área para realizar el mantenimiento de los usuarios de la aplicación**

Implementar los mantenimientos correspondientes a seguridad en un área exclusiva para usuarios con el rol “Admin”.

