عنوان: **رمزنگاری خودکار فیلدهای مخفی در ASP.NET MVC**
نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۴/۱۴ ۲۲:۴۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security

جهت نگهداری بعضی از اطلاعات در صفحات کاربر، از فیلدهای مخفی (Hidden Inputs) استفاده می‌کنیم. مشکلی که در این روش وجود دارد این است که اگر این اطلاعات مهم باشند (مانند کلیدها) کاربر می‌تواند توسط ابزارهایی این اطلاعات را تغییر دهد و این مورد مسئله‌ای خطرناک می‌باشد.

راه حل رفع این مسئله‌ی امنیتی، استفاده از یک Html Helper جهت رمزنگاری این فیلد مخفی در مرورگر کاربر و رمز گشایی آن هنگام Post شدن سمت سرور می‌باشد.

برای رسیدن به این هدف یک Understanding and Extending Controller Factory in MVC (Controller Factory) سفارشی را جهت دستیابی به مقادیر فرم ارسالی، قبل از استفاده در Actionها و به همراه کلاس‌های زیر ایجاد کردیم.
کلاس EncryptSettingsProvider :

```
public interface IEncryptSettingsProvider
    {
        byte[] EncryptionKey { get; }
        string EncryptionPrefix { get; }
    }

 public class EncryptSettingsProvider : IEncryptSettingsProvider
    {
        private readonly string _encryptionPrefix;
        private readonly byte[] _encryptionKey;

        public EncryptSettingsProvider()
        {
            //read settings from configuration
            var useHashingString = ConfigurationManager.AppSettings["UseHashingForEncryption"];
            var useHashing = System.String.Compare(useHashingString, "false",
System.StringComparison.OrdinalIgnoreCase) != 0;

            _encryptionPrefix = ConfigurationManager.AppSettings["EncryptionPrefix"];
            if (string.IsNullOrWhiteSpace(_encryptionPrefix))
            {
                _encryptionPrefix = "encryptedHidden_";
            }

            var key = ConfigurationManager.AppSettings["EncryptionKey"];
            if (useHashing)
            {
                var hash = new SHA256Managed();
                _encryptionKey = hash.ComputeHash(Encoding.UTF8.GetBytes(key));
                hash.Clear();
                hash.Dispose();
            }
            else
            {
                _encryptionKey = Encoding.UTF8.GetBytes(key);
            }
        }

        #region ISettingsProvider Members

        public byte[] EncryptionKey
        {
            get
            {
                return _encryptionKey;
            }
        }

        public string EncryptionPrefix
        {
            get { return _encryptionPrefix; }
        }
```

```
        #endregion

    }
```

در این کلاس تنظیمات مربوط به Encryption را بازیابی مینماییم.
EncryptionKey : کلید رمز نگاری میباشد و در فایل Config برنامه ذخیره میباشد.

EncryptionPrefix : پیشوند نام Hidden فیلدها میباشد، این پیشوند برای یافتن Hidden فیلد هایی که رمزنگاری شده اند استفاده میشود. میتوان این فیلد را در فایل Config برنامه ذخیره کرد.

```
<appSettings>
    <add key="EncryptionKey" value="asdjahsdkhaksj dkashdkhak sdhkahsdkha kjsdhkasd"/>
  </appSettings>
```

کلاس RijndaelStringEncrypter :

```
public interface IRijndaelStringEncrypter : IDisposable
    {
        string Encrypt(string value);
        string Decrypt(string value);
    }

 public class RijndaelStringEncrypter : IRijndaelStringEncrypter
    {
        private RijndaelManaged _encryptionProvider;
        private ICryptoTransform _cryptoTransform;
        private readonly byte[] _key;
        private readonly byte[] _iv;

        public RijndaelStringEncrypter(IEncryptSettingsProvider settings, string key)
        {
            _encryptionProvider = new RijndaelManaged();
            var keyBytes = Encoding.UTF8.GetBytes(key);
            var derivedbytes = new Rfc2898DeriveBytes(settings.EncryptionKey, keyBytes, 3);
            _key = derivedbytes.GetBytes(_encryptionProvider.KeySize / 8);
            _iv = derivedbytes.GetBytes(_encryptionProvider.BlockSize / 8);
        }

        #region IEncryptString Members

        public string Encrypt(string value)
        {
            var valueBytes = Encoding.UTF8.GetBytes(value);

            if (_cryptoTransform == null)
            {
                _cryptoTransform = _encryptionProvider.CreateEncryptor(_key, _iv);
            }

            var encryptedBytes = _cryptoTransform.TransformFinalBlock(valueBytes, 0,
valueBytes.Length);
            var encrypted = Convert.ToBase64String(encryptedBytes);

            return encrypted;
        }

        public string Decrypt(string value)
        {
            var valueBytes = Convert.FromBase64String(value);

            if (_cryptoTransform == null)
            {
                _cryptoTransform = _encryptionProvider.CreateDecryptor(_key, _iv);
            }

            var decryptedBytes = _cryptoTransform.TransformFinalBlock(valueBytes, 0,
valueBytes.Length);
            var decrypted = Encoding.UTF8.GetString(decryptedBytes);

            return decrypted;
        }
```

```
        #endregion

        #region IDisposable Members

        public void Dispose()
        {
            if (_cryptoTransform != null)
            {
                _cryptoTransform.Dispose();
                _cryptoTransform = null;
            }

            if (_encryptionProvider != null)
            {
                _encryptionProvider.Clear();
                _encryptionProvider.Dispose();
                _encryptionProvider = null;
            }
        }

        #endregion
    }
```

در این پروژه ، جهت رمزنگاری، از کلاس RijndaelManaged استفاده میکنیم.

```
RijndaelManaged :Accesses the managed version of the Rijndael algorithm

Rijndael :Represents the base class from which all implementations of the Rijndael symmetric encryption
algorithm must inherit
```

متغیر key در سازنده کلاس کلیدی جهت رمزنگاری و رمزگشایی میباشد. این کلید می‌تواند AntiForgeryToken تولیدی در View ها و یا کلیدی باشد که در سیستم خودمان ذخیره سازی می‌کنیم.

در این پروژه از کلید سیستم خودمان استفاده میکنیم.

کلاس ActionKey :

```
public class ActionKey
    {
        public string Area { get; set; }
        public string Controller { get; set; }
        public string Action { get; set; }
        public string ActionKeyValue { get; set; }
    }
```

در اینجا هر View که بخواهد از این فیلد رمزنگاری شده استفاده کند بایستی دارای کلیدی در سیستم باشد.مدل متناظر مورد استفاده را مشاهده می‌نمایید. در این مدل، ActionKeyValue کلیدی جهت رمزنگاری این فیلد مخفی میباشد.

کلاس ActionKeyService :

```
/// <summary>
        /// پیدا کردن کلید متناظر هر ویو.ایجاد کلید جدید در صورت عدم وجود کلید در سیستم
        /// </summary>
        /// <param name="action"></param>
        /// <param name="controller"></param>
        /// <param name="area"></param>
        /// <returns></returns>
        string GetActionKey(string action, string controller, string area = "");

    }
 public class ActionKeyService : IActionKeyService
    {

        private static readonly IList<ActionKey> ActionKeys;

        static ActionKeyService()
        {
            ActionKeys = new List<ActionKey>
            {
                new ActionKey
                {
                    Area = "",
                    Controller = "Product",
                    Action = "dit",
                    ActionKeyValue = "E702E4C2-A3B9-446A-912F-8DAC6B0444BC",
                }
            };
        }

        /// <summary>
        /// پیدا کردن کلید متناظر هر ویو.ایجاد کلید جدید در صورت عدم وجود کلید در سیستم
        /// </summary>
        /// <param name="action"></param>
        /// <param name="controller"></param>
        /// <param name="area"></param>
        /// <returns></returns>
        public string GetActionKey(string action, string controller, string area = "")
        {
            area = area ?? "";
            var actionKey= ActionKeys.FirstOrDefault(a =>
                a.Action.ToLower() == action.ToLower() &&
                a.Controller.ToLower() == controller.ToLower() &&
                a.Area.ToLower() == area.ToLower());
            return actionKey != null ? actionKey.ActionKeyValue : AddActionKey(action, controller,
area);
        }

        /// <summary>
        /// اضافه کردن کلید جدید به سیستم
        /// </summary>
        /// <param name="action"></param>
        /// <param name="controller"></param>
        /// <param name="area"></param>
        /// <returns></returns>
        private string AddActionKey(string action, string controller, string area = "")
        {
            var actionKey = new ActionKey
            {
                Action = action,
                Controller = controller,
                Area = area,
                ActionKeyValue = Guid.NewGuid().ToString()
```

```
            };
            ActionKeys.Add(actionKey);
            return actionKey.ActionKeyValue;
        }

    }
```

جهت بازیابی کلید هر View میباشد. در متد GetActionKey ابتدا بدنبال کلید View درخواستی در منبعی از ActionKeyها میگردیم. اگر این کلید یافت نشد کلیدی برای آن ایجاد میکنیم و نیازی به مقدار دهی آن نمیباشد.

کلاس MvcHtmlHelperExtentions :

```
public static class MvcHtmlHelperExtentions
    {

        public static string GetActionKey(this System.Web.Routing.RequestContext requestContext)
        {
            IActionKeyService actionKeyService = new ActionKeyService();
            var action = requestContext.RouteData.Values["Action"].ToString();
            var controller = requestContext.RouteData.Values["Controller"].ToString();
            var area = requestContext.RouteData.Values["Area"];
            var actionKeyValue = actionKeyService.GetActionKey(
                            action, controller, area != null ? area.ToString() : null);

            return actionKeyValue;
        }

        public static string GetActionKey(this HtmlHelper helper)
        {
            IActionKeyService actionKeyService = new ActionKeyService();
            var action = helper.ViewContext.RouteData.Values["Action"].ToString();
            var controller = helper.ViewContext.RouteData.Values["Controller"].ToString();
            var area = helper.ViewContext.RouteData.Values["Area"];
            var actionKeyValue = actionKeyService.GetActionKey(
                            action, controller, area != null ? area.ToString() : null);

            return actionKeyValue;
        }

    }
```

از این متدهای کمکی جهت بدست آوردن کلیدها استفاده میکنیم.

```
public static string GetActionKey(this System.Web.Routing.RequestContext requestContext)
```

این متد در DefaultControllerFactory جهت بدست آوردن کلید View در زمانیکه میخواهیم اطلاعات را بازیابی کنیم استفاده میشود.

```
public static string GetActionKey(this HtmlHelper helper)
```

از این متد در متدهای کمکی درنظر گرفته جهت ایجاد فیلدهای مخفی رمز نگاری شده، استفاده میکنیم.

کلاس InputExtensions :

```
public static class InputExtensions
    {
        public static MvcHtmlString EncryptedHidden(this HtmlHelper helper, string name, object value)
        {
            if (value == null)
            {
                value = string.Empty;
            }
            var strValue = value.ToString();
```

```
            IEncryptSettingsProvider settings = new EncryptSettingsProvider();
            var encrypter = new RijndaelStringEncrypter(settings, helper.GetActionKey());
            var encryptedValue = encrypter.Encrypt(strValue);
            encrypter.Dispose();

            var encodedValue = helper.Encode(encryptedValue);
            var newName = string.Concat(settings.EncryptionPrefix, name);

            return helper.Hidden(newName, encodedValue);
        }

        public static MvcHtmlString EncryptedHiddenFor<TModel, TProperty>(this HtmlHelper<TModel>
htmlHelper, Expression<Func<TModel, TProperty>> expression)
        {
            var name = ExpressionHelper.GetExpressionText(expression);
            var metadata = ModelMetadata.FromLambdaExpression(expression, htmlHelper.ViewData);
            return EncryptedHidden(htmlHelper, name, metadata.Model);
        }

    }
```

دو helper برای ایجاد فیلد مخفی رمزنگاری شده ایجاد شده است . در ادامه نحوه استفاده از این دو متد الحاقی را در Viewهای برنامه، مشاهده مینمایید.

```
@Html.EncryptedHiddenFor(model => model.Id)
   @Html.EncryptedHidden("Id2","2")
```

کلاس DecryptingControllerFactory :

```
public class DecryptingControllerFactory : DefaultControllerFactory
    {
        private readonly IEncryptSettingsProvider _settings;

        public DecryptingControllerFactory()
        {
            _settings = new EncryptSettingsProvider();
        }

        public override IController CreateController(System.Web.Routing.RequestContext requestContext,
string controllerName)
        {
            var parameters = requestContext.HttpContext.Request.Params;
            var encryptedParamKeys = parameters.AllKeys.Where(x =>
x.StartsWith(_settings.EncryptionPrefix)).ToList();

            IRijndaelStringEncrypter decrypter = null;

            foreach (var key in encryptedParamKeys)
            {
                if (decrypter == null)
                {
                    decrypter = GetDecrypter(requestContext);
                }

                var oldKey = key.Replace(_settings.EncryptionPrefix, string.Empty);
                var oldValue = decrypter.Decrypt(parameters[key]);
                if (requestContext.RouteData.Values[oldKey] != null)
                {
                    if (requestContext.RouteData.Values[oldKey].ToString() != oldValue)
                        throw new ApplicationException("Form values is modified!");
                }
                requestContext.RouteData.Values[oldKey] = oldValue;
            }

            if (decrypter != null)
            {
                decrypter.Dispose();
            }

            return base.CreateController(requestContext, controllerName);
        }

        private IRijndaelStringEncrypter GetDecrypter(System.Web.Routing.RequestContext requestContext)
        {
            var decrypter = new RijndaelStringEncrypter(_settings, requestContext.GetActionKey());
            return decrypter;
```

```
        }
    }
```

از این DefaultControllerFactory جهت رمزگشایی داده‌هایی رمز نگاری شده و بازگرداندن آنها به مقادیر اولیه، در هنگام عملیات PostBack استفاده میشود.
این قسمت از کد

```
if (requestContext.RouteData.Values[oldKey] != null)
                {
                    if (requestContext.RouteData.Values[oldKey].ToString() != oldValue)
                        throw new ApplicationException("Form values is modified!");
                }
```

زمانی استفاده میشود که کلید مد نظر ما در UrlParameterها یافت شود و درصورت مغایرت این پارامتر و فیلد مخفی، یک Exception تولید میشود.
همچنین بایستی این Controller Factory را در Application_Start فایل global.asax.cs برنامه اضافه نماییم.

```
protected void Application_Start()
        {
            ....
            ControllerBuilder.Current.SetControllerFactory(typeof(DecryptingControllerFactory));
        }
```

کدهای پروژه‌ی جاری

TestHiddenEncrypt.7z

***در تکمیل این مقاله میتوان SessionId کاربر یا AntyForgeryToken تولیدی در View را نیز در کلید دخالت داد و در هربار Post شدن اطلاعات این ActionKeyValue مربوط به کاربر جاری را تغییر داد و کلیدها را در بانکهای اطلاعاتی ذخیره نمود.**

مراجع:

Automatic Encryption of Secure Form Field Data

Encrypted Hidden Redux : Let's Get Salty