ASP. NET 窗体身份验证 Sigout 方法的漏洞及对策的研究

黄　政　（浙江大学计算机科学与技术学院 310027）
　　（嘉兴学院信息工程学院 314001）
张泉方　（浙江大学计算机科学与技术学院 310027）
张丽华　（嘉兴学院信息工程学院 314001）

摘要：本文描述了基于 FormAuthentication. SignOut 方法的局限性，并提供了当窗体身份验证的票据被怀有恶意的用户获取时，怎样让身份验证票据容易回应攻击。

关键词：窗体身份验证　身份验证票据　注销

1　引言

ASP. NET 在 System. Web. Security 命名空间提供了支持窗体身份验证类，在. NET Framework 中，身份验证通常有 Windows 身份验证、窗体身份验证、Passport 身份验证和自定义身份验证。这些验证明显的优势在于开发者不必再编写自己的安全架构，而是可以把. NET 安全架构的内置特性加载到自己的应用程序中去。下面就是基于窗体身份验证的例子，它比较短而且容易理解。

```
Void btnLogin_Click( Object Source, EventArgs e)
{
 //判断用户输入的名称和密码是否是信任的
 if ( FormsAuthentication. Authenticate ( txtName. text, txtPassword. text)
 { //重新定向到用户最初请求的 URL，并为此用户创建一个永久的 cookie
 FormsAuthentication. RedirectFromLoginPage (txtName. text, true);
 }
}
```

窗体身份验证通过 Authenticate 方法根据存储在 web. config 文件中的资料对被指定的用户名称和密码进行检查，来确定是否找到相匹配的凭证。如果用户通过验证，RedirectFromLoginPage 方法就会创建一个身份验证票证，对票证进行加密，把它写入 cookie，并把用户重新定向到他们最初请求的 URL。对票证加密可以采用散列 MD5 或 SHA1 的方法。

也许更多的时候，web 应用程序将凭证存储到自己的数据库中，例如 SQL 数据库或者 XML 文件中。

如果建立受密码保护的身份验证票据，就可以防止被恶意的用户攻击。这种密码保护和其他的参数是在 web. config 文件中配置的。下面的配置就描述了一些常用的值。

```
<forms name = "name"
 loginUrl = "url"
 protection = "All | None | Encryption | Validation"
 timeout = "30"
 path = "/"
 requireSSL = "true | false"
 slidingExpiration = "true | false" >
<credentials passwordFormat = "format" / >
</forms >
```

解释一下这些属性的意义。首先，protection 属性是用来加密保护身份验证票据，通过将其设置为 'All'，指定应用程序利用数据验证和加密来保护 Cookie。RequireSSL 属性指定是否需要安全连接来转换身份验证 cookie，设置为 true，则必须使用安全连接来保护用户凭证，ASP. NET 为身份验证 cookie 设置 Ht-

tpCookie. Secure，兼容的浏览器不返回 cookie，并且除非连接使用的是安全套接字层（SSL）。最后，slidingExpiration 指定是否启用弹性过期时间，在单个会话期间，弹性过期时间针对每个请求来重新设定活动身份验证 Cookie 的过期时间。当设置为 true 时，将启用弹性过期时间。其它的值和. NET framework 不同版本的有关安全的属性值在下面的文章中都有描述。

由于过期机制可以注销用户，开发者经常使用这种方式来注销用户，例如，用户单击注销按钮或者注销链接来注销用户，下面就是使用 formsAuthentication 类的 singout 来注销用户的例子。

```
//logOffBtn_Click 事件激发 SignOut 方法
//来使身份验证票据注销
void LogOffBtn_Click( Object sender, EventArgs e)
{
FormsAuthentication. SignOut( );
Status. Text = "You have been successfully logged our from the appliction. ";
}
```

2 Signout 存在的问题

窗体身份验证类在为开发者使用身份验证的方法和函数提供了很好的接口的同时，也出现了设计的缺陷。就像大部分基于身份验证和会话管理机制一样，这种机制很容易重置 cookie，因此也像会话的被盗一样容易受攻击。在这个示例中这种问题由于创建和持续身份验证票据的方法而被加剧了。当建立起一个票据，它就被存储在 cookie 中。在服务器端，没有存储关于这个票据的记录或者状态，甚至没有存储票据期满时间的任何信息。所有的这些信息都编码存储在票据中。因此只要用户的票据是有效的，例如 cookie 的期满时间还没有到期，票据就会被盗取和被误用，甚至会重新启动 IIS 或者因为某种原因而使票据无效。

调用 SignOut 方法仅仅终止了客户端的 cookie，使之无效，例如下面的代码就是 SignOut 方法。如果开发者使用他们以为可以"注销"用户的方法来防止用户的会话被盗用，就会出现安全隐患。因为使用 SignOut 方法注销用户只是让期满时间过期，下面对这个问题做进一步的解释。

```
Public static void SignOut( )
{ FormsAuthentication. Initialize( );
  HttpContext context1 = HttpContex. Current;
  HttpCookie cookie1 = New HttpCookie( FormsAuthentication. FormsCookieName, "");
  cookie1. Path = FormsAuthentication. _ FormsCookiePath;
  cookie1. Expires = new DateTime ( 0x7cf, 10, 12);
  cookie1. Secure = FormsAuthentication. _ RequireSSL;
  context1. Response. Cookies. RemoveCookie ( FormsAuthentication. FormsCookieName);
  context1. Response. Cookies. Add( cookie1);
}
```

3 不正确的配置窗体身份验证导致的风险

下面介绍三种情况能表示 SignOut 方法的漏洞。

3.1 站点脚本侦听者

如果一个用窗体身份验证机制的应用程序容易受跨站点脚本的攻击，恶意的攻击者使用注入脚本来盗取当前合法的登陆用户的身份验证票据，通过 Internet 把它发送到自己的服务器上并且以模拟的方式使用这些票据直到票据的期满时间到期。

3.2 络嗅听者

如果一个易受攻击的应用程序没有使用 SSL 来保护传输安全，cookie 就很容易在网络传输的时候被盗用。一旦攻击者获取了票据，他们就像上面描述的一样来重置票据和 URL，并且盗用合法用户的会话。

3.3 公享计算机场景

如果一个易受攻击的应用程序允许创建永久 cookie，它就会被使用同台机器的其他人误用。攻击者会在身份票据到期前在共享的机器上搜索浏览器和 cookie 缓存来获取身份验证票据并使用它。

4 证明 Signout 方法的漏洞事例

下面通过建立一个应用程序事例来重现 SignOut 方法的缺陷。

（1）Visual Stdio. NET2003 创建一个新的 ASP. NET 应用程序。

（2）将下面的代码加入到 web. config 中，创建窗

体身份验证。

```
<authentication mode = "Forms" >
<forms name = "ASPXUSERDEMO" loginUrl = "login. aspx" protection = "All" timeout = "60" / >
</authentication >
```

(3) webform1. aspx 重命名为 default. aspx,这个页面是用户必须登陆的页面,它是保护整个应用程序资源的首页。

(4) 加入一个叫 Login. aspx 的页面,添加一个用户邮箱地址的文本框 UserEmail 和一个密码文本框 UserPass,一个复选框 PersistCookie,还有一个消息框 Msg,并写入下面的代码,如下所示:

```
void Login_Click( Object sender, EventArgs E) {
if( ( ( UserEmail. Value = = " joe" ) &&( UserPass. Value = " joe" ) ) || ( ( UserEmail. Value = " admin" )
&&( UserPass. Value = = " admin" ) ) ) {
FormsAuthentication. RedirectFromLoginPage ( UserEmail. Value, PersistCookie. Checked) ;
}
else
Msg. Text = " Invalid Credentials: Please try again" ;
}
}
```

(5) 在 default. aspx 页面上添加一个叫注销的按钮,并建立下面的事件:

```
private void Signout_Click( object sender, System. EventArgs e)
{ if( User. Identity. Name = = "" )
{ Reponse. Redirect( " login. aspx" ) ; }
else
{ FormsAuthentication. SignOut( ) ;
Response. Redirect( " login. aspx" ) ;
}
}
```

(6) 向 default. aspx 页面添加一个操作。在这个例子中,我们选择读取一个文件并且把它作为 Mag. text 文本框的内容。就象下面描述的一样。当然你必须保证使用这些代码来确保 ASPNET/Network 服务帐号有访问文件 secret. txt 的权限。在 default. aspx 页面中添加一个管理员页面 – UpdateTitle. aspx。

```
private void Page_Load ( object sender, System. EventArgs e)
{ if( User. Identity. Name = = "" )
{ btnSignIn. Text = " Signin" ; }
string title = " Default" ;
try{
StreamReader sr = new StreamReader( Request. MapPath( " secret. txt" ) ) ;
Title = sr. ReadLine( ) ;
Sr. Close( ) ; }
catch
{ title = " Exception" ; }
Msg. Text = title;
}
```

(7) 添加一个 UpdateTitle. aspx 页面,在此页面上添加一个 btnTitle 按钮和 txtTitle 的文本框。加入下面的代码来完成概念运用的证据。

```
Private void Page_Load ( object sender, System. EventArgs e)
{ if ( User. Identity. Name! = " admin" )
{ Response. Redirect( " login. aspx" ) ; }
}
private void btnTitle_Click( object sender, System. EventArgs e)
{ StreamWriter sw = new StreamWriter ( Request. MapPath( " secret. txt" ) ) ;
Sw. WriteLine( txtTitle. Text) ;
Sw. Close( ) ;
Response. Redirect( " default. aspx" ) ;
}
```

① 浏览 http://localhost/ < dir_name > /default. aspx 页面。

② 用"admin"/"admin"用户和口令登陆。

③ 单击超级连接浏览"administrative resource"。

④ 添加一些内容到文本框中,单击按钮来更新标题。这将会改变主页的标题。

⑤ 单击注销按钮,将会调用 FormsAuthentication. SignOut 方法。

⑥ 打开 Http 协议,向 UpdateTitle. aspx 发出第二

个请求，这个请求在请求主体中应该有一个窗体提交。

⑦ 把请求拖到"Request Builder"中。

⑧ 编辑标题文本框的内容，单击执行。

⑨ 执行步骤 8 后，刷新主页你会看到新的标题。这个标题已经在步骤 9 的时候保存在"secret. txt"文件中，然后又从"secret. txt"文件中恢复。即使管理员用户已经被注销了，这个请求仍然被成功执行。

5 为使用 Signout 方法提供以下几点建议

上面的程序事例使用下面的方法后，就不会出现步骤 9 的问题了。

(1) 在窗体身份验证中添加健壮的密码保护。这包括密码加密和数据完整性的支持，可以使用 SHA1 散列加密。

(2) 确保在传输中使用传输安全保护身份验证票据，这可以通过使用安全套接字 SSL 来保证在网络中的传输的安全，在 web. config 中将 requireSSL 属性设置为真。

(3) 尽可能的缩短票据期满时间。将属性 timeout 设置为一个比较小的值，通过不启用弹性过期时间保证一个固定的票据期满时间。

(4) 保护身份验证票据不被跨站点脚本激活或者被盗。这只能通过执行输入数据的有效验证和使用 HttpOnly 机制来防止窗体身份验证票据被攻击。

(5) 经常使用特殊的票据名称和路径来防止它们错误的传送。

(6) 使用 HTTP 协议模块为被检查为过期的身份验证票据提供服务器备份存储。

表 1 在 Web. config 中窗体身份验证配置的几点建议

设置	ASP. NET1. 1 默认值	ASP. NET2. 0 默认值	建议使用的值	原因
Cookie 模式	不支持	支持	支持	不加 Cookie 的会话会冒很大的风险，因为攻击者在客户端和服务器端使用查询语句通过窗体身份验证。因此所有的请求必须使用 SSL 和不能被缓存。
FormsCookiePath	/	/	保护虚路径	设置尽可能特殊的路径来防止票据被误传。经常把你的站点区分为安全的和不使用 HTTPS 不安全的文件夹。
loginUrl	需要/ 不需要	需要/不需要	保护虚路径	确保重定向的页面是受 SSL 安全套接字层保护的。
RequireSSL	false	false	true	防止窗体身份验证票据在网上传输时没有用 SSL 安全套接字层来保护。
SlidingExpiration	启用	启用	不启用	不启用弹性过期时间，设置一个尽可能短的会话期满时间来确保 Cookie 安全。
Protection	All	All	All	应用程序使用数据验证和加密来保护窗体身份验证票据。此外，确保 Encryption 属性和 Validation 属性设置为 Autogenerated，IsolateApps 属性设置为 true。
Timeout	30	30	10 – 20 或者尽可能的短	通过尽可能的减少期满时间来确保票据不被盗取。

6 结论

使用窗体身份验证利用 FormsAuthentication. SignOut 方法来注销用户时，为了防止窗体身份验证票据被怀有恶意的用户盗取，在配置 webconfig 属性时最好使用表 1 建议的值，这样就可以将风险降为最低。

参考文献

1 ASP. NET 1. 1 入门经典，Chris Uiiman，John Kauffman 著，杨浩译，清华大学出版社，2004. 9。

2 ASP. NET 程序设计，G. Andrew Duthie 著，李万伦、何蕾、赵海译，清华大学出版社，2002. 7。

3 ASP. NET 安全性高级编成，Russ Basiura，Richard Conway 著，王晓娜、黄开枝译，清华大学出版社，2003. 4。

4 ASP. NET 安全应用程序开发，微软公司著，清华大学出版社，2003. 8。

5 http://support. microsoft. com/? kbid = 900111

6 http://msdn. microsoft. com/library/default. asp? url = /library/en - us/dnpag2/html/tmwa. asp