文章编号:1674-2869(2008)03-0090-04

基于Internet的ASP.NET应用程序安全模型研究

刘黎志,刘 军
(武汉工程大学计算机科学与工程学院,湖北 武汉 430074)

摘 要:提出了一个能解决基于Internet的ASP.NET应用程序安全问题的模型,对用户凭据安全及验证、基于角色的用户授权、安全数据通讯及受信子系统模型等问题作了较详细的描述.

关键词:凭据安全;凭据验证;用户授权;受信子系统模型
中图分类号:TP 309 文献标识码:A

0 引 言

基于Internet的应用程序往往面对复杂的安全问题,具体表现不同用途的应用程序对安全的要求是不一样的[1,2].有些应用程序只是用于发布信息,可以说不存在安全问题,而有些基于电子商务的应用程序则需要对在网络中传输的所有信息进行严格保密.我们所讨论的安全问题主要是指如何保证用户登录系统的凭据安全及保证数据在网络传输中的机密性和完整性.作为Web应用程序的开发者,其目标是建立一个扩展性好,高性能并且安全的应用程序.

1 基于Internet的ASP.NET应用程序的安全性模型

ASP.NET使用两种资源访问模型:受信子系统模型和模拟/代理模型.

受信子系统模型使用固定的系统身份访问系统资源.用户登录到Web服务器后,ASP.NET应用程序首先使用特定的用户身份验证机制验证用户是否为合法用户(Windows验证方式或窗体验证方式),然后使用基于角色的授权机制对用户所能访问的资源进行限制.具有系统资源访问权限的用户被映射到固定的一个系统身份上,ASP.NET应用程序使用该身份访问系统资源.这个固定的系统身份可以是权限较高的'SYSTEM'身份,可以是权限较低的'ASPNET'身份,也可以是自定义的身份.重点讨论默认的'ASPNET'身份,因为它被默认的配置为'以最小的权限'运行应用程序,这符合建立安全的应用程序的要求.

模拟/代理模型将登录到Web服务器的用户身份直接作为访问系统资源的身份.实现该模型需要Web服务器、数据库服务器、客户端操作系统均为Windows操作系统且客户端使用IE浏览器.IIS必须设定使用集成Windows验证或基本验证方式,集成Windows验证使用Kerberos方式,NTLM方式不支持代理.

若在Web服务器和数据库服务器之间建立信任关系,则ASP.NET应用程序使用一个固定的身份访问数据库,这将充分利用.NET数据访问框架所提供的连接池功能,从而显著提高应用程序的可扩展性和效率.与基于Intranet的B/S应用不一样,基于Internet的应用程序不能保证客户端的浏览器一定是IE,所以ASP.NET应用程序对客户端不能使用基于Windows的用户凭据验证方式,而只能使用自定义的窗体验证方式.因此提出以下的安全模型,如图1所示.

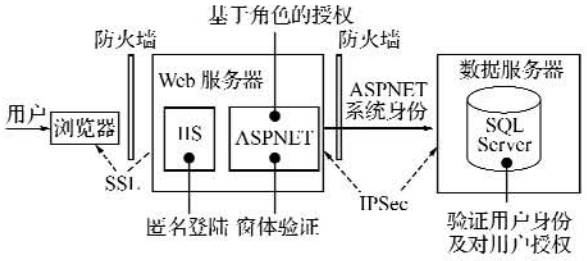

图1 基于Internet的ASP.NET应用程序的安全性模型

Fig. 1 ASP.NET application security model based on internet

从以下几个方面来讨论基于Internet的ASP.NET应用程序的安全问题:(1)保证凭据安全及使用合理的验证机制对用户进行验证.(2)限制合法的用户对系统资源的访问(对用户授权).(3)保证数据在网络传输过程中的安全.

收稿日期:2007-04-12

作者简介:刘黎志(1973-),男,湖北武汉人,讲师,硕士.主要研究方向:商业智能、数据仓库及数据挖掘,基于网络的存储系统。

2　凭据的安全及验证

基于 Internet 的 ASP.NET 应用程序需要保证两类凭据的安全：用户用于登录应用程序系统的凭据及用于访问数据库的凭据. 由于使用窗体验证方式，ASP.NET 应用程序需要提供自定义的用户凭据的验证方式，而用于访问数据库的凭据的验证方式由 SQL Server 提供，我们只需要考虑登录数据库凭据的安全.

2.1　登录应用程序系统的凭据安全及验证

窗体验证方式要求用户在进入系统时必须在自定义的登录页面中提供合法的凭据，若用户凭据通过验证，则将加密后的用户凭据以 cookie 的方式写回客户端. 在用户浏览器与 Web 服务器的会话期间，该 cookie 会附加在用户发出每个 HTTP 请求中表示该用户的合法身份. 用于加密 cookie 的密钥及用于验证 cookie 完整性的密钥由系统自动产生，因此可以认为 cookie 在会话期间是安全的[3].

但需要考虑的是用户第一次向系统提供凭据时，如何保证凭据的安全，因为此时用户的凭据并不是以加密 cookie 的方式在网络中传输，而是以明文的形式. 因此必须对自定义的登录页面申请 SSL 保护，可以根据实际情况决定是否需要客户端证书，从而保证用户凭据的安全. 另外在客户端浏览器和 Web 服务器会话结束后，表示用户凭据的 cookie 要被删除，以防止客户不经验证进入系统. 通常在登录逻辑中设定写回到客户端的 cookie 为临时 cookie 或在用户退出系统时清除客户端的 cookie，以此保证客户端表示用户凭据的 cookie 被删除.

用户的凭据通常保存在数据库中，用来验证凭据的手段通常是比较用户提供用户名和密码和在数据库中的值是否一致. 按这种方式用户的密码要么以明文的形式存放在数据库中，这显然是不安全的. 或者，用户密码经某种加密算法加密后存入数据库，这样做的安全性要比明文方式好，但必须保证加密密钥的安全. 我们使用哈希算法来存储和验证用户凭据，这种方式的最大优点是不必在数据库中保存用户的密码，算法描述如下：

```
/* 用户注册时 */
RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();//生成强随机数
byte[] buffer = new byte[64]; rng.GetBytes(buffer);
//连接强随机数和用户密码
string saltPwd = string.Concat(用户密码, Convert.ToBase64String(buffer));
//用 sha1 哈希算法生成 saltPwd 字符串的哈希值
string hashPwd = FormsAuthentication.HashPasswordForStoringInConfigFile(saltPwd, "sha1");
将用户 ID、强随机数、哈希值存入数据库;
/* 验证用户时 */
根据用户 ID 查询用户 ID 所对应的强随机数和哈希值;
SHA1CryptoServiceProvider sha = new SHA1CryptoServiceProvider();
if(sha.ComputeHash(用户 ID 所对应的强随机数 + 用户提供的密码) == 用户 ID 所对应的哈希值)
    用户通过验证;
else 用户为非法用户.
```

2.2　访问数据库的凭据安全

使用受信子系统模型只需要保证 ASPNET 身份的凭据安全. SQL Server 提供两种用户身份验证方式，Windows 验证方式和 SQL Server 验证方式. 若 Web 服务器和 SQL Server 服务器在一个域中，则可使用 Windows 验证方式. 使用 Windows 验证方式首先需要将 ASPNET 系统身份的密码改为某个已知的密码，然后在 SQL Server 上建一个相同的 ASPNET 镜像身份. 这样，在数据库连接字符串中就不必给出用户名及密码，而只需指定 Trusted_Connection = Yes，从而保证了凭据的安全. 若 SQL Server 服务器要求使用基于 SQL Server 的验证，则连接字符串中必须以明文给出连接数据库的用户名和密码，因此必须加密该字符串[4].

使用 DPAPI(Win32 Data Protection API)对字符串加密. 使用 DPAPI 的最大好处是不需要维护加密密钥，DPAPI 在 Window 2000 操作系统的支持下，利用用户配置文件(user profile)导出加密密钥，密钥由操作系统维护，只有加密数据的用户才能对数据解密. 由于 ASPNET 系统用户没有用户配置文件，所以当使用受信子系统模型时必须定义一个特定的用户调用 DPAPI 对数据库连接字符串进行加密及解密. 给出以下解决方案：

(1) 定义一个特定的用户用于调有加密、解密组件，并建立该用户的配置文件

(2) 生成一个 com+ 组件(由受控代码生成)，其中封装 DPAPI 加密、解密方法. 注册该组件并

且定义该组件只能由1中定义的特定用户启动.

(3) 生成一个Windows服务程序启动加密、解密组件,并将其启动方式由'手动'该为'自动'.理由是Window服务程序可以保证使用加密、解密组件的用户的配置文件的自动装入.

(4) 在ASP.NET应用程序调用加密、解密组件对数据库连接字符串进行加密、解密使用该方式的加密、解密过程如图2所示.

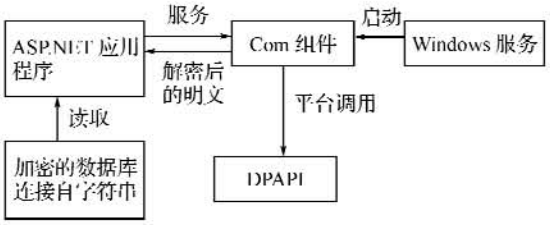

图2 DPAPI加密、解密过程

Fig. 2 Encrypt and decrypt process of DPAPI

3 用户的授权

使用基于角色的授权方式对用户所能访问的资源进行限制.角色实际是对系统功能的划分,使用户成为某个角色的一员,就是赋予用户访问某个系统功能的权限[5].

3.1 ASP.NET应用程序中的用户授权

由于ASP.NET应用程序使用窗体验证方式,因此,可以自定义角色及角色和用户之间的映射关系.可以使用ASP.NET提供的URL授权方式对用户授权,也可以使用程序逻辑对用户授权,重点讨论后者.实现程序逻辑对用户权限的控制,首先定义一个实现IPrincipal接口的类,该类从GenericPrincipal类继承.

```
public class CustomPrincipal:GenericPrincipal
{
  public CustomPrincipal (IIdentity identity, string[] roles):base(identity,roles) {}
  //根据需要扩充类的方法,如验证用户是否同时属于多个角色等等
  public bool IsInAllRole(params object[] roles) {}…
}
```

然后在应用程序的全局文件中实现以下逻辑.

```
public class Global : System.Web.HttpApplication
{
  protected void Application_AuthenticateRequest(Object sender, EventArgs e)
  {  //得到表示用户凭据的加密cookie
    string cookeName = FormsAuthentication.FormsCookieName;
    HttpCookie authCookie = Context.Request.Cookies[cookeName];
    //若表示用户凭据的加密cookie为空,表示用户没有成功登录系统
    if(authCookie == null)
      return;
    else
    {  string[] roles;
      if (Context.Cache[User.Identity.Name] == null)
      {string[] roles = 查询用户所属的角色;
      //缓存用户所属的角色,并定义缓存在最后一次访问的30分钟后过期
      Context.Cache.Insert(User.Identity.Name, roles, null, DateTime.MaxValue, TimeSpan.FromMinutes(30));}
      else
        roles = (string[]) Context.Cache[User.Identity.Name];
      CustomPrincipal principal = new CustomPrincipal(User.Identity,roles);
      //将自定义的CustomPrincipal对象赋予当前HTTP请求的Principal对象
      Context.User = principal;}
    }
}
```

最后在应用程序页面的后台代码中调用自定义的CustomPrincipal类中相应方法确定当前用户对某个系统功能有无权限.例如判断用户同时属于"Manager"和"Admin"这两个角色才能执行某个方法的代码为:

```
public void Method()
{
  if (((CustomPrincipal) this.User).IsInAllRole("Manager","Admin"))
    {代码段;}
  else return;
}
```

3.2 数据库服务器中的用户授权

由于SQL Server信任ASP.NET应用程序,因此ASP.NET应用程序以固定的ASPNET身份访问数据库.给予ASPNET身份所有应用程序所需的数据库资源的操作权限显然是不安全的,虽然这是一个简单有效的方法.若要实现基于角色的授权,可在应用程序需要访问的数据库中建

立多个应用程序角色，并分别赋予不同的数据库访问权限. 在 ASP. NET 应用程序中根据用户所属的角色分别调用 sp_setapprole 系统存储过程激活相应的应用程序角色，ASPNET 身份实际是在同时模拟多个应用程序角色，从而实现对数据库资源的基于角色的授权. 调用 sp_setapprole 系统存储过程需要以明文方式给出应用程序角色名及密码，因此存放密码前必须对密码加密. 可以调用封装 DPAPI 的 com$^+$ 组件对密码加密，应用程序在激活应用程序角色时再将密码解密后传递给 sp_setapprole系统存储过程.

4　数据在网络传输中的安全

使用 SSL(Secure Sockets layer)保证客户端浏览器与 Web 服务器之间所传输的数据的机密性及完整性. SSL 在客户浏览器与 Web 服务器之间建立一条加密的数据通道，从而保证数据传输的安全. 可以根据实际情况决定是否需要客户端证书，从而保证只有受信任的客户才能访问 Web 服务器中的重要资源. 为保证 ASP. NET 应用程序的执行效率，应只对承载敏感数据的页面申请 SSL 保护. IPSec(Internet Protocol Security)提供在传输层的安全通讯解决方案，可以保证两个主机间的数据传输安全. 数据库服务器和 Web 服务器根据具体的情况，通过合理的配置本地的安全策略实现安全的数据传输.

5　结　语

ASP. NET 是统一的 Web 开发平台，用来提供开发人员生成企业级 Web 应用程序所需的服务. 对于 Web 开发人员来说，保证 Web 站点的安全是一个关键而又复杂的问题. 安全的系统需要仔细地规划，而且 Web 站点管理员和程序员必须清楚地了解有关保证他们站点安全的选项. ASP. NET 与 Microsoft . NET 框架及 IIS 协同工作以提供 Web 应用程序安全性. 在应用 ASP. NET 开发基于 B/S 模式的 Web 应用时，在安全性方面作了一些探讨，并在实践中加以了应用.

参考文献：

[1] 贺金凌. 基于. NET 平台的 Web 解决方案的安全机制[J]. 计算机应用与软件，2002，19(11)：19-21.

[2] 杨　强，卢建军. 一种基于. NET 的安全服务决策平台的具体实现[J]. 计算机应用，2002，22(9)：83-84.

[3] 陈志刚. ASP. NET 安全特性在 Web 中的应用. 电脑与信息技术[J]，2004，12(3)：53-57.

[4] 谭冠正. ASP. NET 中认证安全特征评述. 沈阳工业大学学报[J]，2003，25(3)：250-254

[5] 黄益民，杨子江，平玲娣. 安全管理系统中基于角色访问控制的实施方法[J]. 浙江大学学报，2004，38(4)：408-413.

Research of ASP. NET application security model based on internet

LIU Li-zhi , LIU Jun

(School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430074, China)

Abstract: This paper introduces a model that can solve security problems of ASP. NET application based on Internet. We mainly describe credential security & authentication, authorization based on role, secure communication and trusted subsystem model.

Key words: credential security; credential authentication; authorization; trusted subsystem model

本文编辑：陈晓苹