基于 ASP. NET 的 B/S 架构下的项目管理系统的网络安全模式设计

张文涛[1] 常红星[2]
(中国科学技术大学信息科学技术学院 合肥 230027)[1]
(中国科学院自动化研究所综合信息系统研究中心 北京 100080)[2]

摘 要 本文介绍了基于 ASP. NET 平台 B/S 架构的项目管理系统网络安全的设计,分析了 ASP. NET 的三层网络结构的安全机制,给出了通过身份验证、权限控制、数据加密、存储过程访问数据库等手段实现系统的安全性的技术细节。
关键词 ASP. NET,B/S 架构,身份验证,权限,存储过程

Model Design of a Network Security System for Project Management Systems Based on the B/S Architecture in ASP. NET Platform

ZHANG Wen-Tao[1] CHANG Hong-Xing[2]
(College of Information Science & Technology, University of Science & Technology, Hefei 230027)[1]
(Institute of Automation, Chinese Academy of Science, Beijing 100080)[2]

Abstract This paper introduces the design of a network security system for the project management systems bases on B/S framework in ASP. NET platform. It analyzes the security mechanism of the 3-tier network architecture of ASP. NET. The paper presents the detailed technologies to achieve the system security by the methods of identity validation, access control, data encryption, and access database by storage procedures.
Keywords ASP. NET, B/S framework, Identity validation, Access right, Storage procedure

1 引言

随着 Internet 技术的发展和基于 Web 开发平台的推出,目前在项目管理系统中 B/S 结构逐渐取代了传统的 C/S 结构,成为了主流结构,得到了广泛的应用。

B/S 结构是一种以 HTTP 为传输协议,客户端通过浏览器访问 Web 服务器以及与之相连的后台数据库的体系结构。B/S 构架具有良好的跨平台性、可扩展性和易更新升级等优点[1],正是 B/S 架构的这种开放性的特点,也对网络系统的安全体系的设计和实现提出了新的要求,就是要保证合法用户对系统资源的安全访问,同时又防止非法访问者的入侵。

本文以 CMM 项目管理系统的开发为例,阐述了在 ASP. NET 平台下开发的 B/S 结构的项目管理系统的安全模式设计,从系统架构、角色及用户权限控制、数据库访问等方面分析了如何保障数据和系统的安全。

2 基于 ASP. NET 的三层网络架构的安全性

系统基于 Windows2000 Server 操作系统,采用 ASP. NET 实现 Web 服务器与数据库的连接,后台数据库为 SQL Server 2000 系统,以 Vistual Studio. net 为系统开发平台,系统采用 B/S 的三层架构。

系统利用 ASP. NET 部署 B/S 的三层架构,三层是由显示层、中间层、数据层组成。显示层就是利用浏览器为客户提供应用服务的图形界面,负责直接跟用户进行交互;中间层位于显示层和数据层之间,由应用服务器和 Web 服务器实现系统的业务逻辑功能;数据层是三层中的最底层,负责数据的存储和访问。每层的功能非常清晰,层与层之间不能跨越,客户端不直接与数据库进行交互,而是通过 COM/DCOM 通讯与中间层建立连接,中间层经过 ADO. NET 实现对数据层的数据进行访问,实现了显示、数据、逻辑的分开,减少了耦合度,更加灵活,适于维护。

ASP. NET 在网页中使用基于事件的处理,显示层的页面代码和后台的代码分离,系统采用 C# 作为后台代码的语言。. NET 中可以方便地实现组件的装配,后台代码通过命名控件可以方便地使用自己定义的组件。显示层放在 ASP 页面中,数据库操作和业务逻辑用组件来实现,这样就很方便地实现了三层架构。这样的结构减少了入口点,减少了由于客户端被破坏而给数据库带来损失的风险,保证了系统的安全。图 1 所示为系统的网络架构。

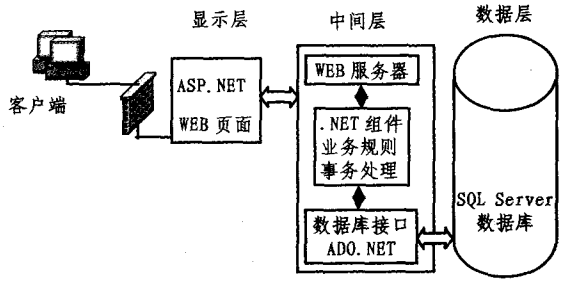

图 1 系统的网络架构

3 ASP. NET 应用的安全性

在系统中 ASP. Net 与 Microsoft Internet 信息服务(IIS)

张文涛 硕士,主要从事网络应用开发;常红星 研究员,研究方向为智能信息处理。

联合起来协同工作提供了优秀的安全控制[3]，包括身份验证和权限控制两部分。身份验证就是对发送请求信息的用户进行身份识别，比如用户名和密码，决定他的访问权限，如果身份被核实，那么就说明用户通过了验证。一旦验证通过，权限控制程序就会决定用户是否有权限访问他所调用的资源。

3.1 身份验证

ASP.NET 支持三种类型的身份验证，分别是 Windows 身份验证、表单(Forms)身份验证、Passport 身份验证。本系统采用基于表单的验证，是三种验证方式中最灵活的一种。

基于表单的验证服务使用 Cookies 来验证用户，所谓 Cookie 就是一小段文本信息，伴随着用户请求和页面在 Web 服务器和浏览器之间传递，用户每次访问站点时，Web 应用程序都可以读取 Cookie 包含的信息。在登录页面中，用户输入他的凭证(用户名和密码)，提交该页面送回服务器，应用程序根据存储在数据库中的数据来验证请求，用户通过验证后，ASP.NET 发出一个 Cookie，里面包括了为此用户产生的一个有效的身份票据[2]，该身份票据是用户进行系统访问的"通行证"。在随后发出的请求页面的请求报头里包含此 Cookie，如果没有通过验证则用户被重新定位到登录页面。身份票据的确定可以根据应用需求而设计，本系统中身份票据 Http-Cookie 由用户名和用户标识号构成。图 2 显示了系统身份验证的过程。

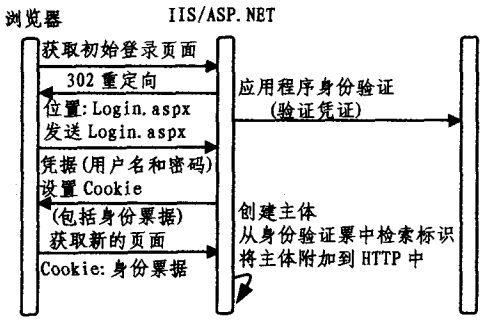

图 2 系统身份验证过程

将 ASP.NET 配置为使用表单的身份验证需要修改 Web.config 文件，并且可以通过此文件指定录面。在 Web.Config 文件中对〈authentication 〉的标签修改如下：

```
〈authentication mode="Forms" 〉//选择表单验证方式
〈forms name="MyAppFormAuth" loginUrl="login.aspx" protection="All" timeout="20" path="/"〉//login.aspx 是登录页面
〈/forms〉
〈/authentication〉
```

3.2 权限控制

本系统的权限控制采用基于角色的访问控制，权限赋予角色，角色分配给用户。一个用户可拥有多个角色，一个角色可授权给多个用户。用户不直接与权限相连，而是通过所属的角色享有权限，实现了用户与访问权限的逻辑分离，极大地方便了权限管理。一般来讲角色数要少于用户数，这样可以通过更改角色的权限实现对大批量用户权限的更新，不必一个一个地设定用户的权限，从而降低了管理的成本。

本系统有多个功能模块，可以根据需要设定多个角色，定制角色的权限就是设定每个角色针对不同功能的权限，对每个功能都包括五种权限：浏览权限、查询权限、添加权限、修改权限、删除权限。分别代表可以对此功能页面下的数据执行浏览、查询、添加、修改、删除操作。角色的权限设定后只需给用户分配角色即可，同一个用户可属于多种角色，不过他对某一功能的操作权限是取此用户所属几个角色中对此功能最高的权限。系统可以根据企业的情况任意的添加、删除、修改角色，定制角色的权限，通过这种角色的权限定义可以实现各种复杂的安全策略。图 3 显示系统的用户、角色、功能、权限之间的关系。图 4 显示的是系统角色权限设定的界面。

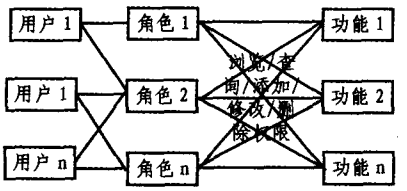

图 3 系统的用户、角色、功能、权限的关系图

角色权限设定: 角色 1 权限设定

功能	权限				
功能 1	浏览	查询	添加	修改	☑ 删除
功能 2	浏览	查询	添加	☑ 修改	☑ 删除
功能 3	浏览	查询	添加	修改	删除
功能 4	浏览	查询	☑ 添加	☑ 修改	☑ 删除
功能 5	浏览	查询	添加	修改	☑ 删除
功能 6	☑ 浏览	☑ 查询	☑ 添加	☑ 修改	☑ 删除
...			...		
功能 n	浏览	查询	添加	修改	☑ 删除

图 4 系统角色权限设定的界面

通过权限控制，保障了合法用户顺利实现系统功能，禁止了非授权用户对系统的入侵。在显示页面的时候，根据用户所属角色的不同，显示不同的页面。在登录页面中，用户输入凭证用户名和密码，系统验证通过后，根据输入的用户名得到所属的角色，根据角色和功能权限的对应关系进而得到此用户对所有功能模块的权限情况，系统据此加载功能链接区页面发送给浏览器，用户就看到了属于自己的主页面。在此页面上只显示出此用户有权限访问的链接，他无权访问的链接不显示。图 5 显示根据用户权限定制网页的流程。

系统通过用户名-密码的检查来验证用户的身份，保证合法的用户才能登录系统，登录系统后又根据用户对各模块功能权限的不同定制出属于此用户可访问的页面链接，从模块功能的链接上保证了安全。系统进一步地在用户执行每一次操作，比如查询、添加、修改、删除时，都要进行许可验证，只有当用户所属角色具有对目标数据操作许可的情况下，才能通过认证执行相应的操作。否则系统发出权限不足提示。这样的权限控制严格地保护了数据的安全，保障了系统的可靠运行。系统权限控制的工作流程如图 6 所示。

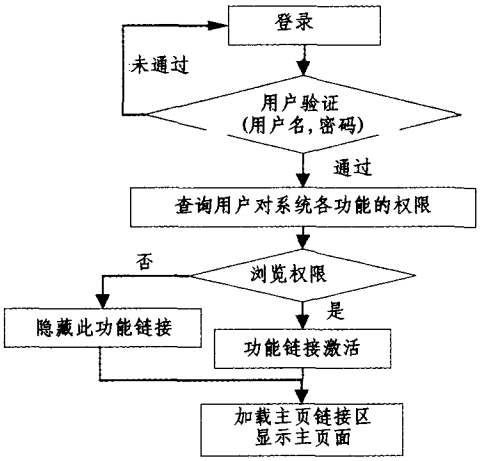

图 5　根据用户权限定制网页的流程

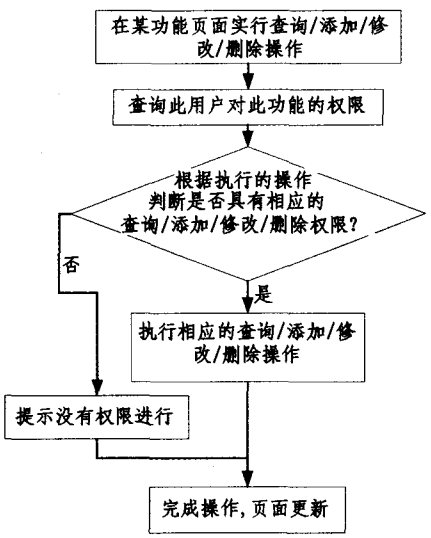

图 6　用户执行操作的权限控制

4　数据库访问的安全性

4.1　数据库中用户口令的加密处理

在基于网路的系统中,用户的信息安全是非常重要的,一旦用户的密码被盗取后,系统的数据和整个数据库也就不安全了,通过对用户的口令加密可以解决这个问题,用户密码不以明文的方式存储在数据库中,而是存储它们加密后的版本。当我们需要对用户进行验证时,只是对用户的口令再进行加密,然后把它与数据库中的加密口令进行比较即可。

ASP. NET 可以方便地实现对密码的加密,在命名空间 System. Web. Security 中包含了类 FormAuthentication,其中有一个方法 HashPasswordForStoringInConfigFile 可解决此问题,它支持用于加密字符串的"SHA1"和"MD5"散列算法[3],将用户提供的密码变更乱码,然后存储起来。这样在数据库中存储的就不是实际的密码而是加密后的密码,保证了用户、数据和数据库的安全。

4.2　ADO. NET 访问数据库

本系统采用的是三层架构体系,用户不直接访问数据库,而是通过中间层的 ADO. NET 在. NET 平台中提供对数据库的访问服务。ADO. NET 是全新的数据访问接口,它把访问数据和操纵数据彻底隔离,可以在非联机状态下使用。

ADO. NET 用两个核心组件来访问和处理数据:. NET 数据提供程序和 DataSet。数据提供程序是一组包括 Connection、Command、DataReader 和 DataAdapter 对象在内的组件[4],是专门为数据处理以及快速地只进、只读访问数据而设计的组件。DataSet 就是数据集,是把数据库中的数据映射到内存缓存中的所构成的数据容器。在我们的系统中数据库用的是 SQL Server,数据提供程序就是 SQL Server 数据提供程序 SQLServer. NET。

通过 SQL Server 数据提供程序完成和数据库的连接并将数据填充到 DataSet 对象,然后客户端再通过读取 DataSet 来获得需要的数据。这种结构如图 7 所示。

图 7　通过 ADO. NET 访问数据库的结构

4.3　使用存储过程的安全性

系统通过中间层的组件 ADO. NET 访问数据库,在对数据库的访问上,本系统全部采用存储过程实现对数据的操作。通过存储过程可以实现用户和数据的隔离,避免了任何对数据表的完全访问和更新,而是通过执行存储过程实现对数据表的操作。存储过程还可以帮助你逃离代码的安全隐患,保护你免受一种叫做'SQL 注射'的攻击,这种攻击主要会在你的合法输入参数中添加如 AND 或者 OR 的操作符[5]。存储过程还有助于对用户屏蔽数据库的内部实现以减少你的应用程序泄密的危险,保证了数据的正确和安全。

相对于直接使用 SQL 语句,在应用程序中直接调用存储过程还有以下好处:存储过程可以将多个 SQL 语句打包在一个存储过程中,以批处理的方式执行,而不是从客户端发送多个请求,这样就减少了网络通信量;执行的速度更快,存储过程在调用时只进行一次解析和优化,再次调用这个存储过程时就可以直接从内存中调用即可,而执行 SQL 语句时每次都要进行解析和优化;存储过程还可以使页面完全独立于数据库中表的具体实现,只要在存储过程接口不变的情况下开发人员对数据库的任何改动都不需要对显示层和中间层的程序作改动,这样大大方便了系统的修改和维护。

在使用 ADO. NET 组件调用存储过程时,为了确保数据安全性,程序中调用存储过程语句中全部使用参数而不是把具体的值代入。以下代码显示了系统使用参数传递值的一个示例。

```
SqlCommand cm = new SqlCommand("CheckLinkRight");// Check-
    LinkRight 为存储过程名
cm. CommandType = CommandType. StoredProcedure;

// 参数设置
cm. Parameters. Add(new SqlParameter("@User_ID", SqlDbType.
    Int));
cm. Parameters ["@User_ID"]. Value = int. Parse (User _ ID.
    Text);
// User_ID 为此存储过程中的一个输入参数

cm. Parameters. Add (new SqlParameter ("@ LinkType", SqlDb-
    Type. VarChar));
cm. Parameters ["@LinkType"]. Value = LinkType. Text;
// LinkType 为此存储过程中的另一个输入参数
```

而不用下面的方法把值直接带入来执行存储过程

```
Sql ="exec dbo. CheckLinkRight " + int. Parse (User_ID.
    Text) + "," + LinkType. Text + "'"
```

通过使用存储过程不仅对开发人员编码和系统的修改维护提供了方便,更重要的是对数据的安全和系统的安全运行

(下转第 108 页)

随机数 k_{i1} 和 k_{i2} 给投标者 B_i，其中 $k_{i1}, k_{i2} \in \{0,1\}^l$，$l$ 是安全参数。A_1 和 A_2 将收到的数据和一次性随机数 k_{i1} 和 k_{i2} 存入本地数据库。

投标：假设每个投标者的投标价不同，投标者 B_i 选取他的秘密标价 $b_i \in \{1,2,\cdots,p\}$ 并冗余编码为

$\beta_{i1} = Encode(b_i, A_1) = (x_{i1}, \cdots, x_{ip})$ 和

$\beta_{i2} = Encode(b_i, A_2) = (y_{i1}, \cdots, y_{ip})$

其中 $x_{ij}, y_{ij} \in_R \{0,1\}$ 使得当时 $j = b_i$，$x_{ij} \oplus y_{ij} = 1$；当 $j \neq b_i$ 时，$x_{ij} \oplus y_{ij} = 0$。

投标者 B_i 选择一次性随机比特串 $r_{i1}, r_{i2} \in \{0,1\}^l$，发送 $c_{i1} = h(\beta_{i1} \| r_{i1} \oplus k_{i1} \| i \| ID_{i1})$ 和 $c_{i2} = h(\beta_{i2} \| r_{i2} \oplus k_{i2} \| i \| ID_{i2})$ 到公告牌 BB 作为对投标的承诺。再通过安全信道向拍卖行 A_1 和 A_2 分别发送 $(\beta_{i1} \| r_{i1} \| i \| ID_{i1})$ 和 $(\beta_{i2} \| r_{i2} \| i \| ID_{i2})$。$A_1$ 和 A_2 首先验证注册中心的签字，然后从本地数据库中取出 k_{i1} 和 k_{i2} 并验证 $c_{i1} = h(\beta_{i1} \| r_{i1} \oplus k_{i1} \| i \| ID_{i1})$ 和 $c_{i2} = h(\beta_{i2} \| r_{i2} \oplus k_{i2} \| i \| ID_{i2})$ 是否成立，接受通过所有验证的投标为合法标书，对使用过的 k_{i1} 和 k_{i2} 进行标记。

开标：① 确定中标价（M+1 价）。当投标期结束后，执行下面的算法，利用 A_1 和 A_2 的交互将求出中标价（M+1 价）。

(1) $s_i := 0$，$k_i := p+1$；

(2) $k_i := k-1$；

A_1 计算并发送 $X_k = (x_{1k} \| \cdots \| x_{nk})^t \bmod N$ 到公告牌 BB，A_2 计算并发送 $Y_k = (y_{1k} \| \cdots \| y_{nk})^t \bmod N$ 到公告牌 BB；

(3) 如果 $X_k \neq Y_k (\Leftrightarrow (x_{1k} \| \cdots \| x_{nk}) \neq (y_{1k} \| \cdots \| y_{nk}))$，则 $s_i := s+1$；

(4) 如果 s=M+1，则输出中标价为 k 并终止程序；否则转(2)。

②确定中标人。利用 A_1 和 A_2 的交互将确定出中标人。

对于投标者 B_i，A_1 计算并发送 $M_i = (x_{i(k+1)} \| \cdots \| x_{ip})^t \bmod N$ 到公告牌 BB，A_2 计算并发送 $N_i = (y_{i(k+1)} \| \cdots \| y_{ip})^t \bmod N$ 到公告牌 BB。

如果 $M_i \neq N_i (\Leftrightarrow (x_{i(k+1)} \| \cdots \| x_{ip}) \neq (y_{i(k+1)} \| \cdots \| y_{i(k+1)}))$，说明投标者 B_i 的投的标价 b_i 大于 k，则投标者 B_i 中标，否则投标者 B_i 没有中标。

3 协议分析

投标者身份的匿名性：由于拍卖行 A_1 和 A_2 只知道临时身份，对于其他投标者和外部攻击者，这种临时身份也不知道，因此投标者的身份是匿名的。

不可否认性：任何投标者不能否认所投的标书，因为标书中包含了他或她的秘密临时身份，通过与注册中心合作，可以找到恶意的参与者。

标价的秘密性：由于 $(\beta_{i1} \| r_{i1} \| i \| ID_{i1})$ 和 $(\beta_{i2} \| r_{i2} \| i \| ID_{i2})$ 与 b_i 是相互独立的，任意单个的拍卖行不可能知道投标者所投标价，投标者与任意一个拍卖行勾结也不可能知道其他投标者所投标价，除非两个拍卖行勾结。由于 RSA 模数 N 分解未知，从 $X_k = (x_{1k} \| \cdots \| x_{nk})^t \bmod N$ 不可能知道 $(x_{1k} \| \cdots \| x_{nk})$，同样从 $Y_k = (y_{1k} \| \cdots \| y_{nk})^t \bmod N$ 中也不可能知道 $(y_{1k} \| \cdots \| y_{nk})$。因此，即使开标后，上述结论仍然成立，协议满足标价的秘密性。

不可伪造性：一份完整的标书包括发送到公告牌的承诺和发送给两个拍卖行的数据，其中包含了投标者临时秘密身份和一次性随机数的信息。由于除注册中心以外的任何人不可能事先知道投标者的临时秘密身份，除拍卖行以外没有人事先知道一次性随机数，因此任何人不可能伪造一份合法的标书。

抗重放攻击：由于拍卖行对使用过的一次性随机数 k_{i1} 和 k_{i2} 进行了标记，攻击者在计算上不可能重放以前拍卖活动中合法的标书而不被发现。

高效性：执行开标算法至多需要 p 轮交互，至多 $2p \log_2 t$ 次模乘法运算，与参与者的数量无关，故拍卖方案是高效的、实用的。

结论 文中提出了一种新的基于 RSA 函数的 M+1 电子拍卖方案，可以保护投标者的匿名身份，任何投标者不能否认所投的标书，未中标价不被泄露，伪造一份合法的标书或重放以前拍卖活动中合法的标书而不被发现，在计算上都是不可能的。该方案执行开标算法在最坏的情况下只需 p 轮交互，$2p \log_2 t$ 次模乘法运算，与投标者的数量无关，协议安全、高效。

参 考 文 献

1 Franklin M, Reither M. The Design and Implemention of a Secure Auction Service [J]. IEEE Trans on Software Engineering, 1996, 22(5): 302～312

2 Kikuchi H, Harkavy M, Tyger D. Multi-round Anonymous Auction Protocols [C]. In: Proceedings of the First IEEE Workshop on Dependable and Real-time E-commerce System. Berlin: Springer Verlag, 1998. 62～69

3 Kudo M. Secure Electronic Sealed-bid Auction Protocol with Public Key Cryptography [J]. IEICE Trans on Fundamental, 1998, E81-A(1): 20～27

4 Mu Y, Varadharajan V. An Internet Anonymous Auction Scheme [C]. Lecture Notes in Computer Science, Berlin: Springer Verlag, 2001. 171～182

5 杨加喜，王育民. 一种安全高效的 M+1 电子拍卖[J]. 网络安全技术与应用，2006,11:87～89

6 Wurman P R, Walsh W E, Wellman M P. Flexible double auction for electronic commerce: Theory and Implementation, Decision Support System, 24. 17～27 [DB/OL]. http://www.csc.ncsu.edu/faculty/wurman/papers/Wurman-Dss-98.pdf,2002-05-15

(上接第 103 页)

提供了有力的保证。

总结 本文对采用基于 ASP.NET 构建的 B/S 结构的项目管理系统的安全模式设计进行了讨论，这种基于 Web 的应用系统的安全体系设计相对于传统的 C/S 模式，既要考虑数据访问的安全，又要考虑网络的安全。Web 开发平台 ASP.NET 的安全访问机制使得 Web 的安全性和可靠性得到了基本的保障，采用 B/S 的三层架构，将显示层、中间层、数据层在逻辑上相互独立，减少了耦合度，保证了系统的安全。采用了身份验证、权限控制、数据加密、存储过程访问数据库等多种安全策略保证了数据的安全，限制了网络的入侵，最终保障了网络系统的安全

参 考 文 献

1 李兰友，杨晓光编著. ASP.NET 实用程序设计[M]. 清华大学出版社，2005

2 邹显春，张为群. 一种主动网络管理系统结构的分析与研究[J]. 计算机科学，2006(10)

3 李敏波. ASP.NET 1.1 高级编程[M]. 清华大学出版社，2005

4 (美)Basiura R，等著. 王晓娜，黄开枝译. ASP.NET 安全性高级编程[M]. 清华大学出版社，2003

5 王珺，王崇骏，谢俊元，陈世福. 基于 Agent 的网络入侵检测技术的研究[J]. 计算机科学，2006(12)