.NET Security Programming

.NET 安全编程

(美) Donis Marshall 著
余 波 张立浩 译

清华大学出版社

.NET 安全编程

(美) Donis Marshall 著

余波 张立浩 译

清华大学出版社

北 京

内 容 简 介

Windows 和基于 Web 的环境下的安全性一直都是人们关注的问题，特别是在破坏安全的活动日益猖獗的今天。而.NET 安全性正是解决长期以来困扰计算机产业的安全问题的新方法，它与 Win32 安全性截然不同。

本书主要讲述了.NET 安全编程，同时引入了许多新的术语、概念及语法。内容涉及.NET 体系结构、运行库安全策略、代码访问安全性、基于角色的安全、ASP.NET 安全性、密码术，以及定制.NET 安全性，本书还分别介绍了 System.Security 命名空间和 System.Security.Permissions 命名空间。

本书适合在.NET 平台下工作的专业开发人员。

Donis Marshall: .NET Security Programming

EISBN:0-471-22285-2

北京市版权局著作权合同登记号：01-2003-5609

图书在版编目(CIP)数据

.NET 安全编程/(美) 马歇尔 著；余波，张立浩 译. — 北京：清华大学出版社，2003

书名原文：.NET Security Programming

ISBN 7-302-07252-3

I. N… II. ①马… ②余… ③张… III. 计算机网络—程序设计 IV. TP393

中国版本图书馆 CIP 数据核字(2003)第 082866 号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦

http://www.tup.com.cn　　邮　　编：100084

社 总 机：010-62770175　　客户服务：010-62776969

组稿编辑：曹康

文稿编辑：于平

封面设计：康博

版式设计：康博

印 刷 者：北京鑫海金澳胶印有限公司

发 行 者：新华书店总店北京发行所

开　　本：185×260　印张：15.75　字数：403 千字

版　　次：2003 年 10 月第 1 版　2003 年 10 月第 1 次印刷

书　　号：ISBN 7-302-07252-3/TP・5269

印　　数：1～4000

定　　价：35.00 元

前　言

Scott Stabbert(Microsoft 的一位工程师)第一个向我介绍了.NET 安全编程。Scott 将.NET 安全性描绘成与原来的 Win32 安全性完全不同。他用“完全不同”这个词直奔主题。听完他介绍了.NET 安全性的复杂性后，我觉得仅用该词语来概括是远远不够的。. NET 不使用安全属性，安全描述符，Discretionary Access Control List(自由访问控制列表)，System Access Control List(系统访问控制列表)，访问控制项，NTLM 以及传统 Win32 安全性的其他功能。以前的安全模型已经完全被弃用。在 Win32 安全 API 上投入了相当多的精力后，这一转变使我怅然若失。

.NET 安全性是解决长久困扰计算机产业的安全问题的一个新途径。Scott 先生对.NET 安全性的狂热也激发了我对.NET 的兴趣。对.NET 安全性作了单独的研究后，它同样给我留下了深刻印象。在这之后不久，我开始在 Microsoft 公司和其他公司教授.NET 安全编程，从而也就自然而然地编写了这本关于.NET 安全性的书。

.NET 安全性的基本前提就是代码(而不是用户)才是安全策略的核心。这一简单而又根本的转变解决了长期以来的安全问题，如共享安全上下文和引诱攻击。我们每个人都曾经收到过一些值得怀疑的邮件——邮件发送者的姓名难以辨认，主题也是含混不清。更糟糕的是，这类邮件还带有附件。不过，对于这种附件把戏，任何一个有理智的人都不会打开来历不明的附件。然而，有时只是打开邮件就已经足以引发一次攻击了。正因为如此，您才会坐在那里反复考虑要不要打开那些可疑邮件。给嵌入的代码指派一个独立的安全上下文将使这一问题迎刃而解。而对于一些级别较低的代码，例如附件，最好授予有限的权限，防止引发安全攻击。现在，您可以打开邮件甚至附件，而不需要担心除垃圾邮件以外的任何可能产生的后果。

如何给程序集指派一个安全上下文呢?在 Win32 操作系统中，最为普遍的验证模式是 Challenge/Response 验证。challenge 就是一个已知用户的口令。信任机构根据 challenge 本地计算该口令，如果相同，用户就通过验证并被分配一个访问令牌，该令牌确定了该用户的安全上下文。response 是发送到请求机器的访问令牌。.NET 对于代码采用一个类似的模型。公共语言运行库(CLR)询问程序集的证据而不是口令。CLR 是一个信任机构。根据所提交的代表该程序集的证据，评估访问的信任级别并授予合适的权限。例如，下载缓存中运行的代码被部分信任并授予有限的权限。

尽管我们反对这么做，而实际上每个人仍然以管理员特权在本地机器上登录。

在本地计算机上运行的程序将会继承您的安全上下文。这是一个“共享安全”的上下文问题。您机器上的任何一个程序都为管理者权限所信任吗？回答是“不”。.NET 访问独立于用户上下文的代码，并且根据应用程序(而不是用户)的特征指派一个信任级别和安全上下文。用户的安全上下文仍由操作系统强制。不过，.NET 为代码增加了另一个安全层。因此可以说.NET 安全性很完美。

0.1 托管语言

托管语言包括 C#、Visual Basic.NET、JSharp.NET、JScript.NET 等，而且这一队伍仍在扩大之中。在不久的将来，将会有 Perl.NET、Fortran.NET 和 Ada.NET，并且可能会出现惊世的 COBOL.NET。

本书使用 C#作为托管语言，其实选择哪种语言并不重要。如果您是一位 Visual Basic 或 Java 编程人员，可以阅读 Framework Class Library(FCL)中的.NET 安全类，这在所有.NET 语言中都是相同的。

看一下下列同一安全程序的 C#和 Visual Basic.NET 版本。C#版本的程序如下所示：

```
// C# Security Code
using System;
using System.Security;
using System.Security.Permissions;
class Starter
{
 public static void Main()
{
    FuncA();
    StreamWriter s=new StreamWriter(@ " c:\test.txt " ,true);
    s.WriteLine( " Test " );
    s.Close();
}
void FuncA()
{
    FileIOPermission p =
            new FileIOPermission(FileIOPermissionAccess.Read,
                @ " C:\test.txt " );
        p.PermitOnly();
 }
}
```

下面是该程序的 Visual Basic.NET 版本。

```
' VB Security Code
imports System
imports System.Security
imports System.Security.Permissions
imports System.IO
public class Starter
    public shared sub Main()
    FuncA()
    dim s as new StreamWriter( " c:\test.txt " , true)
    s.WriteLine( " VB Test " )
```

```
        s.Close()
    end sub
    public shared sub FuncA()
        dim p as new _
      FileIOPermission(FileIOPermissionAccess.Read, _
                        " C:\test.txt " )
        p.PermitOnly()
    end sub
end class
```

这些程序几乎是完全相同的。最大的区别在于C#是区分大小写的，并且要求每个语句的末尾都加分号。Microsoft 公司的总经理及首席软件设计师 Bill Gates 将托管语言的选择称为生活方式的选择。而 C#最适合我的生活方式。

本书的 Visual Basic.NET 示例代码均可在 Web 站点 www.wiley.com/compbooks 中找到。

本书中需要特别注意的安全问题将在安全警告和注意中作说明。

0.2 章节概述

本书的目的在于让专业开发人员全面认识安全编程。其中包括：代码访问安全性、基于角色的安全、密码术和定制.NET 安全性。个别主题(如密码术和 ASP.NET 安全性)仅用一章的篇幅难以讲述清楚。因此，对于这些主题，我们只是希望读者对之能有一个基本的理解。

和任何一种新技术一样，.NET 引入了大量的新术语、概念和语法。幸运的是， Michael Dunner(他也是微软公司的工程师)提供了很大的帮助。.NET 安全性的语法并不十分困难，而新概念和术语的理解可能更具有挑战性。考虑到这个原因，我们将把重点放在.NET 安全性的这些方面。

0.2.1 .NET 体系结构

对.NET Framework 有一个基本理解有助于学习.NET 安全编程。第 1 章将探讨.NET 体系结构、托管语言和公共语言运行库(CLR)，并解释.NET 的常用术语和概念。还会介绍 Web Forms、ASP.NET、XML Web 服务、ADO.NET 和.NET remoting。CLR 是.NET 的引擎。本章讲解了 CLR 的服务，例如实时编译(Just-in-Time Compilation)和无用单元收集(Garbage Collection)。本章最后介绍了一个基本的 C#应用程序。

0.2.2 .NET 安全性的核心概念

本书的目标是为应用程序构建一个安全边界(perimeter)。第 2 章讨论了此边界中有关验证的一些小技巧。Manifest(清单) 和代码验证证实了程序集的正确性。然后回顾了应用程序域。应用程序域属于“轻量级”的进程，被提供和单独的进程相同的保护。强名程序集和共享的程序集比私有程序集更安全。强名程序集的优点会作详细讨论。从 Internet 和内部网下载的代码在下载缓存中执行并被授予有限的权限。第 2 章中还将讨论这个边界的其他更多内容。

0.2.3　运行库安全策略

运行库安全策略(Runtime Security Policy)相当于.NET 安全的路标。它把程序集映射到安全权限。CLR 使用运行库安全策略确定程序集的信任级别，并在加载时将正确的权限授予程序集。在加载时授予程序集的权限是最多的。第 3 章讲述了运行库安全策略的组件：代码组、权限集、权限和证据。.NET 提供 Microsoft Framework Configuration 和 Caspol 工具作为管理运行库安全策略的方法。第 3 章演示了这两种工具。

0.2.4　代码访问安全性

代码访问安全性强制了运行库安全策略。主要的命令是 Demand，该命令执行堆栈遍历以便确认调用者已经请求了权限。请求一个与敏感资源相关的权限，从而保护对这一资源的访问。代码访问安全性也允许开发人员注释并改善运行库安全策略。开发人员可以命名可选的权限，指派最小的权限集，并且拒绝不需要的权限。第 4 章提供了对代码访问安全性的指导。

0.2.5　基于角色的安全

.NET 并没有完全忘记用户。基于角色的安全服务于一般用户和 Windows 用户。一般身份和负责者不是 Windows 用户。相反地，Windows 身份和负责者是已经经过身份验证的 Windows 用户(用 NTLM，NT Lem Manager)。Windows 负责者是访问令牌的包装器。角色是组，但不一定是 Windows 组。第 5 章将介绍如何分配一个执行安全上下文的线程，如何去模拟一个 Windows 用户，以及如何在企业应用程序中管理角色。

0.2.6　ASP.NET 安全性

ASP.NET 和 Web Forms 是.NET 中必不可少的组成部分，即具有高度可用性、可从任何平台访问的、可部署到各种设备中的软件。当无线技术成为占优势的标准时，ASP.NET 以优秀的桌面出现。由于特征是不相同的，因此保护 Web 应用程序与保护客户端应用程序的方式也有显著的区别。Web 应用程序在公共“场所”运行，会受到无数的安全攻击。第 6 章阐述了 ASP.NET 应用程序独特的安全需求，包括窗体验证，URL 授权和基于角色的安全等。

0.2.7　密码术

密码术是关于保密和证明身份的。在深入具体内容之前必须理解密码术的结构。第 7 章首先介绍了重要的密码术概念和术语，说明了加密、解密、散列和数字签名。该章首先用大量示例代码讲述了 Crypto API，最后详细讨论了.NET 中的密码术，特别介绍了 CryptoStream 类。

0.2.8　定制.NET 安全性

Microsoft 使.NET 具备可扩展性。新的攻击持续不断地出现，同时也会出现抵御这些攻击的算法。在安全性这一领域，任何事物都不是静止的。另外，每个应用程序都是惟一的，有着各自独特的需求。Microsoft 不可能预见安全需求的每一次转变。解决的方法是使.NET 安全性具备可扩展性。第 8 章将处理在运行库安全策略中出现的定制目标。在.NET 中可以很容易地定制权限、权限集、代码组和证据。

0.2.9　System.Security 命名空间

System.Security 命名空间包含了.NET 安全性的基础结构的大部分内容。了解该命名空间的知识是了解.NET 安全性的第一步。第 9 章介绍了该命名空间的类并且提供了示例代码。

0.2.10　System.Security.Permissions 命名空间

System.Security.Permissions 命名空间包括很多运行库安全策略和代码访问安全性所使用的类。第 10 章介绍了这些类并且对运行库安全策略、代码访问安全性和定制.NET 安全在这章中作了补充说明。

目　　录

第1章 .NET体系结构

.NET 安全性在技术上不是孤立的，它是.NET Framework 整体的一部分。因此，对于.NET Framework 的基本理解是尝试进行.NET 安全编程的前提条件。本章展示了.NET Framework 体系结构与编程的基本概念。这仅仅是一个概述，它并不能取代想要完全掌握该主题所需的独立学习(如果想从一个开发者的角度了解关于.NET Framework 的全面讨论，作者推荐以下这本书：《.NET Framework Essentials》，作者是 Thun L. Thai 和 Hoang Q. Lam，由 O'Reilly & Associates 于 2002 年 2 月出版)。

Microsoft .NET 并不是 Win32 操作模型的一个变体。而且，尽管存在着一些负面的报道，它也没有图谋不轨地伪装成 Java 的样子。如果您只是简单地将其与现有的产品进行比较，那么您永远也不可能理解或充分地解释.NET 的含义。.NET 是全新的。.NET 引入了关于计算软件和设备的全新的操作模型与观点。

那么.NET 与 Java 有相似之处吗？还有，它与 Win32 有相似之处吗？是的，确实有一些相似的地方，但是它们之间存在着更多的区别。要想成功地使用.NET 编程，您就必须乐于接受这项新的技术，并完全理解那些使.NET 与众不同的技术。当面向对象语言刚刚面世的时候，开发人员们也面临过类似的挑战。遗憾的是，这似乎是一种固定的模式。当时，许多编程人员很快就学会了那些面向对象语言的语法，并将他们的用 C 编写的应用程序移植到 C++或 SmallTalk。然而，如果对于面向对象编程缺少必要的理解，那么这些新的应用程序就只是由面向对象语言的语法编织而成的过程式程序。有一些开发人员除了语法之外，还花时间学习了面向对象程序设计的原理和目的。结果，他们编出的应用程序就是真正的面向对象程序，并且提供了这种新的编程模型所能预见的所有好处。同样地，理解.NET 的原理与体系结构对于创建能够提供新型解决方案的应用程序来说，也是绝对必要的。某些业界分析家断言，Microsoft 已经在.NET 上将整个公司作为赌注。作者并不这么认为。.NET 确实代表了一笔巨大的投资。然而，Microsoft 作为一家多元化的、拥有数十亿美元的公司，它开发出了很多产品，并且在软件业的众多领域内都占有着相当大的市场份额。而且，Microsoft 已不仅仅是一家软件公司，它已经将业务扩展到其传统强项以外的多个市场中。

但是，认识到 Microsoft 没有在名为.NET 的悬崖上蹒跚而行并没有降低.NET 的重要性。.NET 确实代表了产品开发中的新理念。通过.NET，将会出现一个全新的产品系列，它们将在接下来的 5 到 10 年内把 Microsoft 的销售推进到一个新的高度。即使.NET 的推出失败了，或者更有可能的是被缓慢地采纳，Microsoft 也能恢复过来并继续生存下去，虽然它可能要变得黯淡一些。重要的是，.NET 使得 Microsoft 摆脱了在 Windows 上进退维谷的局面。虽然 Windows 取得了巨大的成功，但它仍然是一个框(box)。.NET 帮助 Microsoft 从这个框中解脱了出来，并能够为所有用户开发应用程序。这个新的机遇不仅仅为 Microsoft，而且为各地的开发者都注入了成长的动力。

几年前，作者参加了在佛罗里达的奥兰多召开的专业开发人员会议，并出席了 Microsoft .NET 的正式发布会。当时，比尔·盖茨是主要的发言人。他的演讲包含了一段有趣的视频。视频将.NET 描绘为一个新的标准，该标准可以让软件在任何地点、任何时间、任何平台、以及各种大小的设备上都能运行。

任何地点：这项特性已经被多次报道过了，但它仍是值得一提的："Microsoft 曾经在意识到 Internet 的重要性方面迟了一步，只是后来才匆忙接受它的"。最近，Microsoft 已经开始弥补过去的迟缓，而.NET 正是其在该过程中采取的一个重大举措。Internet 不是.NET 的附件，而是与该产品无缝地交织在一起。Internet 被规划、集成并实现在.NET 中，这包括了对像 XML 和 HTTP 这样的开放标准的采纳。从本质上说，任何提供了能够理解 XML 或 HTML 的浏览器的平台都是一个潜在的.NET 客户机。

任何时间：Internet 是每周 7 天，每天 24 小时这样一直开放的，它从不关闭。由于.NET 支持 Internet，所以.NET 应用程序(例如 Web 服务)也完全能在任何时候被用户访问。

任何平台：.NET 是一个多语言、多平台的操作环境。与单语言、多平台的 Java 相比，.NET 提供了 C#、Visual Basic .NET、以及其他多种与.NET 兼容的语言。在.NET 中编程不需要学习一种全新的语言，而针对 Java 虚拟机(JVM)编程却需要学习 Java 语言。对许多人来说，这是一个很大的障碍。公共语言运行库(Common Language Runtime)对于所有.NET 语言来说都是通用的。此外，Microsoft 还发布了公共语言基础结构(Common Language Infrastructure，CLI)文档，它包含了一系列针对任何平台(例如 Linux)创建.NET 公共语言运行库所需遵守的准则。如果您想对此有了解，请访问 www.go-mono.com。未来的开发人员将可以在 Windows 平台上创建.NET 应用程序，而在 Linux、Unix、Macintosh 或其他任何一个提供了公共语言运行库的平台上运行它们。

各种大小的设备：.NET 标志了 Microsoft 第一次对开放标准的广泛支持，虽然它并不十分热心。Microsoft 采纳了 HTTP、SMTP、SOAP、XML 以及其他很多标准。这就意味着任何支持这些标准的设备都能够主动地参与到.NET 会话中。这将会解放个人数字助理(PDA)、手持设备以及嵌入式设备。这些设备本身缺少运行强大应用程序的能力，例如技术上已成熟的 Microsoft Office。而通过使用这些开放式标准，这些设备将可以借助后端服务器的能力，运行几乎任何程序。设想一下，您冰箱内部的嵌入式芯片能够远程访问 Microsoft Word，列出食品清单，然后再将清单提交给网络打印机打印出来。具有文字处理能力的冰箱——这真的很酷！

1.1　.NET 的 Web 能力

Microsoft .NET 被赋予了 Web 的能力。开发人员能够使用 ASP.NET、XML Web 服务和 ADO.NET 轻松地创建功能丰富的 Web 应用程序。这代表了一个 n 层企业应用程序中的上层、中层，与底层。尽管如此，您还是不要相信有关 Microsoft 已经放弃了客户端应用程序这样的花言巧语——有一些应用程序永远也不会适于作为服务器端操作。Windows Forms—— 一种新的窗体生成引擎，以及其他一些.NET Framework 中的附件，使得传统 Windows 应用程序的开发变得更为直观，并向其中添加了额外的特征。

1.2　.NET 组件

.NET 引入了一种新的组件模型，它在很大程度上是隐含的，而 COM(Component Object Model，组件对象模型)中的混乱情况则被去除了。在.NET 中，开发人员使用标准语言的语法来创建、发布和导出组件，而无需学习另外的什么内容。.NET 解决了 COM 的很多不足之处，包括易于陷入 DLL Hell、语言的不兼容性、引用计数等。

在 API 层上进行 COM 编码是一项艰巨的任务。众多的接口(例如 IUnknown、IDispatch、IConnectionPoint 等)和系统函数(例如 CoGetClassObject 和 CoFreeUnusedLibraries)代表了一条巨大的学习曲线。虽然 MFC(Microsoft Foundation Classes，Microsoft 基础类)、ATL(Active Template Library，活动模板库)和 Visual Basic 减轻了其中的一些负担，但它们针对不同的语言提供了用于创建 COM 对象的不同解决方案。

如何在.NET 中创建组件呢？在服务器应用程序中，您可以使用您喜欢的一种.NET 语言的语法，来定义一个公共类。而在客户机上，您需要导入一个对组件应用程序的引用，然后再创建一个该组件类的实例。仅此而已。随后，您就可以使用该组件，而 COM 中晦涩的语法则消失了。

COM 对象的生存期由开发人员来控制。AddRef 和 Release 这两个方法已经声名狼藉。它们是 IUnknown 接口的方法，被用来控制 COM 对象的持久性。如果它们没有被正确地实现，那么一个 COM 对象可能就会被过早地释放；或者，相反地，永远也不会被释放并导致内存泄漏。.NET 内存管理器(也被称为无用单元收集器)负责为应用程序管理组件的生存期，从而不再需要 AddRef 和 Release。

COM 在很大程度上是一个 Windows 标准，并且它与其他平台中的组件沟通也十分困难。但 COM 仍然是一项值得的尝试，而且它确实推进了组件开发的概念。然而，现在火炬已经交到了.NET 的手中。由于使用了开放式标准，.NET 组件潜在地可以被每个人在任何时间访问。

组件的版本控制是一个与 COM 有关的大问题，它也促成了 DLL Hell。而另外一个促成因素就是 COM 对 Windows 注册表的依赖。让我们假设，同一个进程内服务器(COM DLL)的两个不同版本被安装在了同一台计算机上，且新的版本被首先安装。在这种情况下，旧的组件就会覆盖掉新版组件在注册表中的设置，从而使得客户端被重定向到错误的版本上。而需要使用更新的服务的客户端就会立即中断。.NET 没有依靠注册表来登记组件，这就减少了 DLL Hell 出现的可能性。

作者已经教授开发人员.NET 一段时间了，他通常向他的学生提出下面这个问题来作为课程的开始：有谁可以描述一下.NET 给最终用户带来的好处？学生们很快就提到了公共语言运行库、ASP.NET、XML 集成、无用单元收集器等。这些都是很好的功能，但它们所带来的好处都是面向编程人员的，而不是面向最终用户。而.NET 给用户带来的好处就是.NET 所引入的新一代软件，这些软件在任何地点、任何时间、任何平台、以及各种大小的设备上都能够运行。