PROGRAMMER TO PROGRAMMER™

Visual Basic .NET
Code Security Handbook

Visual Basic .NET
代码安全手册

Eric Lippert 著
徐燕华 崔伟 译

清华大学出版社

Visual Basic .NET 代码安全手册

Eric Lippert 著

徐燕华 崔伟 译

清华大学出版社

北 京

北京市版权局著作权合同登记号：01-2002-3188

内 容 简 介

本书讲述了如何运用 Visual Basic .NET 来确保代码安全的问题。首先讨论了应用程序安全，常见的攻击类型以及攻击者的动机。然后简单介绍了基于角色的 Windows 安全系统和基于代码的.NET 安全系统，接着进一步探讨了.NET 安全的高级主题，例如链接请求、继承请求等。最后通过大量的例子介绍了如何编写安全和非安全的代码，如何发现安全故障。

本书适用于所有 VB.NET 编程人员，本书所论述的有关安全体系的内容适用于所有.NET 语言，本书可以帮助开发人员编写出安全、高效、健壮的代码。

EISBN：1-86100-747-7

Visual Basic .NET Code Security Handbook

Eric Lippert

图书在版编目(CIP)数据

Visual Basic .NET 代码安全手册/(美)林普特著；徐燕华，崔伟译.—北京：清华大学出版社，2003

书名原文：Visual Basic .NET Code Security Handbook

ISBN 7-302-06360-5

Ⅰ. V… Ⅱ. ①林…②徐…③崔 Ⅲ. BASIC 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2003)第 010102 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)

http://www.tup.com.cn

责任编辑：王军

封面设计：康博

印 刷 者：北京牛山世兴印刷厂

发 行 者：新华书店总店北京发行所

开　　本：787×1092 1/16　**印张**：16　**字数**：332 千字

版　　次：2003 年 3 月第 1 版　2003 年 3 月第 1 次印刷

书　　号：ISBN 7-302-06360-5/TP・4801

印　　数：0001～4000

定　　价：35.00 元

前　言

如今，.NET 语言可以使程序员明确地表示安全概念，因为它已经将安全编程概念作为.NET Framework 的一部分。.NET Framework 提供了丰富的安全类，为编写安全代码提供了相应的保证。

通过在.NET Framework 的类中实现这些概念，.NET 运行库使得所有.NET 编程语言都可以使用这些新的安全性能。尽管本书着重介绍的是 Visual Basic.NET，但书中讲述的有关安全系统的信息也适用于其他.NET 语言。

0.1　本书读者对象

本书针对的是那些希望编写健壮、安全的代码的 Visual Basic .NET (VB.NET)编程人员。由于几乎没有哪个开发人员希望编写脆弱的、不安全的代码，因此它适用于所有 VB.NET 编程人员！

过去，大部分 Visual Basic 编程人员使用 VB 实现 4 个目的：

- 使用 VBA 为 Office 文档(和其他应用程序)添加丰富的后台功能
- 实现基于窗体的应用程序
- 实现用于 Web 页面或其他应用程序的组件
- 使用 VBScript 在 Active Server Page(ASP)上实现服务器端功能

上述每种开发都有与安全相关的不同方面。例如，Web 页面可用控件的开发人员必须确保有恶意的 Web 页面不可以使用 VB 控件删除客户机中的文件或是访问未经授权的信息。ASP 页面上服务器端脚本的开发人员必须确保有恶意的客户程序不能在服务器上访问未经授权的信息。.NET 提供了一个强大的新安全模型，如果使用得当，那么可以确保所有类型的应用程序的安全。

本书假定读者已很熟悉 VB.NET 语言、基于对象的编程和.NET Framework。尽管如此，我们还会在必要的时候讲述这些概念。

0.2　本书主要内容

第 1 章——应用程序安全性：概述了安全问题，讨论了可能会对代码发起攻击的类型和攻击者的动机。

第 2 章——.NET 和安全性概述：本章介绍了基于代码和角色的安全概念，分别将它们与.NET 和 Windows 安全系统联系起来。我们讲述了理解.NET 安全系统需要的关键性概念，例如托管堆栈，但本章大部分内容讲述的都是有关.NET 基于代码的安全模型。我们特别介绍了安全策略是如何根据证据应用于代码，然后确定授予的权限。

第 3 章——高级的.NET 安全性：我们深入介绍了.NET 安全系统的一些更复杂的功能，例如链接请求、继承请求和在.NET 中使用基本角色的安全。

第 4 章——可定制的安全应用程序：我们介绍了可定制的安全应用程序，讨论了定制证据和定制策略。

第 5 章——如何会导致编写不安全的代码：在本章和下一章中，我们将讨论特定的编码方式(好的和坏的)。我们将显示如何发现常见的和少见的弱点，如何修复它们以及如何在第一时间避免出现此类问题。本章将集中讲述产生弱点的“最差的实例”。

第 6 章——如何使代码安全：本章讲述了“最好的实例”，可以帮助防止第 5 章中遇到的问题。本章最后提供了一个便捷的总结最好和最坏实例的检验表。

第 7 章——发现安全故障：将本书介绍的知识应用于实际生活中，发现自己或别人代码中的安全故障很重要。本章运用了一些简短、但满是安全漏洞的实际方法，试图让读者找到其中的弱点。

0.3 本书未涉及的内容

本书是为 VB.NET 编程人员提供的一个手册，这些人需要了解.NET 安全系统，需要知道常见的导致弱点的编码错误，希望使用安全系统编写健壮的代码来避免这些弱点。本书将不包括以下内容：

- 大型安全应用程序所固有的结构设计问题。例如：我们将讨论防止有恶意的用户数据侵入 SQL 后端数据库的方法。我们将不讨论如何配置数据库，不准备讨论用于与数据库通信的加密协议，不讨论如何确保事务处理不进行否认，也不讨论如何防止数据包窃听器捕获并重新发送包来欺骗服务器等。
- 各种加密算法的具体实现和它们之间的细微差别。例如，我们将讨论设法保密时所犯的常见错误，但不会深入地进行介绍，例如各种散列算法之间的区别。我们不准备讨论何时使用 Triple DES，以及何时使用 RSA 这样的内容；选择何种算法是个很复杂的问题，这已超出了本书的范围。
- 正确配置 ASP 或 ASP.NET 服务器的详细内容。Web 服务器的身份验证模型大部分使用 Windows 基于用户的安全系统，本书将不详细地进行介绍。我们将简单讨论一下如何在 ASP.NET 中使用基于代码的安全，以及服务器端代码中一些常见的安全错误。我们将不讨论基本身份验证与 Windows 身份验证之间的区别或是其他 ASP 安全配置的内容。

目　　录

第1章　应用程序安全性

本章的目的是打下一个基础，提供一个基本的思维习惯：有恶意的人想对用户不利，通过设计和实现安全方案就可以保护他们。我们将通过提出和解决您们可能会在代码安全性方面碰到的问题来进行介绍。

- 安全性、危害、攻击者和脆弱性等概念的含义
- 谁想攻击我的代码？
- 他们将如何攻击我的代码？
- 我们可以如何去做？

首先还是看一下如今人们开发的应用程序类型，以及为什么在很多情况下，代码安全性得到(或将得到)比以往更多的关注。

1.1　安全性前景

如今所开发的应用程序比传统的“压缩包装”的应用程序组织更为松散。现在的应用程序可以是多层的，在客户机和服务器上运行代码。可以用由不同出处编写的完全不同的部分组成应用程序。可以通过 Internet 在全世界范围内发送数据，在从一台机器发送到另一台机器时跨越“信任边界”。

在以前的单块应用程序时代，几乎不需要考虑代码安全性，因为应用程序的所有部分都是一起编写的。这些应用程序所操作的数据从头到尾都在完全控制之下。开发现代应用程序要求开发人员必须明确使用代码安全概念，如权限和策略。

编写安全应用程序有许多常用的技术，对它们的讨论贯穿于本书各个章节中。在几乎所有情况下，像验证输入是格式良好的或用最低限度的权限运行代码之类的技术都很有用。我们将特别介绍一下现代应用程序的常见情形，例如：

- 为信任您的用户编写一个应用程序，例如开发人员。然而，该应用程序可能调用用户不信任的第三方的“定制”组件。

例如，作为用户，您可能相信 Intenet Explorer 是由 Microsoft 的没有恶意的开发人员编写的，但可信任的应用程序可以下载您并不信任的有恶意的人编写的 Web 页面。同样，您可能相信 Microsoft Excel 是由您所信任的人编写的，但它所运行的宏指令可能包含了有恶意的第三方所编写的病毒。您可能正用 VB.NET 编写类似的应用程序。那么该如何保护用户免受恶意的攻击，而仍能正常工作呢？

- 如果在上面的方案中，应用程序为潜在的恶意的定制代码提供一个可用的对象模型，那么在恶意的调用者调用时，对象模型必须是健壮和安全的。如何限制不可信的代码使用您的对象模型呢？
- 可能您编写了一个用于第三方的应用程序的定制组件，必须考虑用户并不完全信任您的代码的情况。现在问题就变了：如何编写在有限制的安全环境中运行的代码？
- 我们也介绍了一些特定的代码安全性问题，它们是在编写服务器端 ASP.NET 页面时出现的。我们不准备详细介绍 ASP.NET 安全配置，因为这超出了本章范围，但将介绍一些方法来避免带来常见的脆弱性问题。

注意，“安全性方案的设计和实现”意味着不仅仅是要仔细地编写代码。我们不仅要很好地编写安全应用程序，也要使得安装应用程序的管理员可以很容易地部署。安全应用程序使得用户可以很容易地安全运行代码，而不要求他们都是安全专家。

本书的重点是教会开发人员如何编写技术上正确的安全代码。记住，技术正确性只是整个安全性的一部分。最终，开发人员、管理员和最终用户都必须重视安全性。如果管理员作出了较差的决策，或是如果用户允许“社会工程”利用他们，那么所有的技术正确性都毫无作用。

1.2 担心安全性的原因

显然，没有人想编写不安全的、脆弱的代码。真正的问题是“要考虑多少内容？”。遗憾的是，在有最后期限限定的现实生活中，总会发现和修复一些安全性问题，直到最后因为太晚而不能很好地完成工作。

使得安全性变得这么重要有很多原因。首先高度概括一下：

- 第一，最为重要的原因是您有设计出不会危害其他人的代码的义务。人们依赖于他们的应用程序，希望您所提供的工具不会危害他们的想法是完全合理的。在人们和商业活动都依赖于及时访问准确的、秘密的数据的社会中，拒绝服务、窃取和篡改数据都是极其有害的。

遗憾的是，由于普遍存在的安全性问题如电子邮件和宏病毒，我们现在处于这样一个世界中，一些人对打开电子邮件或查看 Web 页面充满恐惧。用户需要知道，他们的日常动作不会导致他们受到伤害。帮助构建一个安全的、可信任的基础结构是每个人的责任。

- 第二，编写不安全的代码会损坏名誉。尽管出于某种原因，您个人不易受到用户攻击者的攻击，但编写易受到攻击的应用程序会对您的名声不利。攻击不仅会感染您的代码，也会对整个平台造成危害，这会减缓该平台的推广，从而减

小了客户数量。

- 第三，编写不安全的代码会直接影响到销售状况。编写不安全代码的名声可能会导致更低的销售额或是无法续订合同。而且在装载代码后，修复所有的缺陷会比事先修复它们的费用昂贵得多。安全脆弱性尤其是这样。如果代码崩溃或不运行，通常都是推迟到下一版本才装载补丁。在发布之前可以为客户找到一个工作区。然而，发布的代码中的严重安全缺陷通常需要立即解决；黑客不会等待下一版本的发布！实现、调试、测试和装载安全补丁的代价是很昂贵的。更糟糕的是，在压力下完成的工作通常都会带来更多的缺陷。另一种选择就是用很“严格”的工作区：让用户禁用主要的功能——假定可以禁用易受攻击的功能！
- 第四，在我们这个法制社会有法律条文。如果您的客户认为您的代码中的不安全性因素是疏忽的结果，那么他们可能会对您提出诉讼。

说明：

事实就是：只要部署了代码，就易受到攻击。因此要在受到攻击前作好准备。

1.3　什么是安全性

为了让开发人员编写安全代码，并将它提供给客户，.NET 运行库提供了一个相当复杂、高度可配置的安全系统。安全系统与是否是房屋的报警系统，银行的金库或是.NET 安全系统都无关，但它们的目的都是一样的：

说明：

任何安全系统的主要目的是防止攻击者蓄意破坏的预谋得逞(至少是能够减缓他们破坏的速度)。

但如何确切描绘攻击者呢？在计算机世界中的危害又是怎样的呢？

1.3.1　恶意企图

安全系统与导致危害的预谋有关；尽管它是阻止突发性危害的安全系统的一个很有用的副作用，安全系统将处理恶意的攻击。

如果您认为没有人想恶意攻击您或您的用户，那么这种想法太天真了；雇员的不满，缺乏职业道德的竞争者从事间谍或破坏活动，有着大量空闲时间的蓄意破坏者们编写病毒，以及罪犯们热衷于访问您的名称、地址和信用卡号。还有一些可能的情况：潜

在攻击者和潜在受害者一样多。

注意，本书将使用术语“攻击者”描述威胁来源。它比常用的“黑客”更准确。

注意：

黑客就是一类只是出于了解的目的而想深入了解一些系统秘密的人。在印象中，黑客就是安全威胁，这意味着未经授权就访问数据的人就是一种威胁，但威胁是各种各样的，并不仅限于普通意义上的黑客。

许多攻击者并不是出于了解系统的目的，而是为了他们自己的需要；他们只是想造成伤害或是想独自获取知识来增强他们造成危害的能力。一些攻击者只是蓄意破坏者，而其他人只是想树立在同行中的威信，例如丑化一个流行的 Web 站点。然而，有一些攻击者的目的完全是想获取金钱。

攻击者来自社会各个阶层。他们可能是老练的、知识渊博的专家，或是运行别人编写的恶意程序的无知“小孩”。攻击者可能是有财团支持的和有组织的，或是单独作战的穷学生。通常，发现安全脆弱性比利用它更难；相反，即使相对没有经验的攻击者也可以利用其他人发现的安全弱点。

最好是将系统设计得富有弹性，可同时抵御老练的、有目的的攻击和无知小孩的强有力的攻击。

说明：

保守一些；不要低估攻击者。如果假定您的攻击者不但狡猾而且人数众多，那么您就更有需要去编写安全代码了。

1. 确定他们不会攻击我！

假定问题不存在(或是其他人的问题)可能是您所犯的最大错误；任何部署过的软件都会受到攻击。然而，影响代码安全有许多因素，包括：

- 软件部署的广度
- 软件处理的值或信息的敏感度
- 使用软件的组织的重要性或知名度

您可能觉得您的项目在这些方面的风险都很小。然而，这意味着您编写代码的努力将不会受到重视，这并不是您想要的结果。

另外，Internet 的流行和在网络环境中工作的软件的相应发展也助长了攻击者的行动。软盘引导扇区上的病毒攻击何时是个尽头呢？攻击者可以从世界的任一角落攻击联网的机器。他们可以极其容易地到处传播恶意软件。

2. 时间对您不利

对所有的攻击者要特别注意的是：他们比您有多得多的空闲时间。攻击者乐于花数

日、数周或数月钻研别人的代码来寻找弱点。和您不同，他们不用面对装载的最后期限，不用参加小组会议，不用作最后一刻的修改。攻击者可以使用同样专业而优秀的工具，寻找代码中的弱点，或者可以编写他们自己的工具。

说明：

举个很有趣的例子，攻击者(尽管这是个无恶意的黑客)创建的工具是 XBox 安全系统硬件部分的一个成功逆向工程。查看 ftp://publications.ai.mit.edu/ai-publications/2002/AIM-2002-008.pdf 可获取详细信息。如果无恶意的黑客可以做到这一点，那么恶意的黑客也可以。

攻击者将安装您的软件，接着监控从注册表和磁盘中写入或读取的每个字节，调用每种操作系统 API 和分配的每个字符串，直到他们知道您的全部秘密。和您的用户不同，他们将阅读文档的全部内容，寻找设计中的弱点。

这就是为什么一开始就要对代码安全进行设计的原因。如果不这样做，攻击者将很容易地找到攻击点来打击您，因为他们有更多的时间和精力去寻找。

1.3.2　信任、安全和意图

不可以信任任何人，也不可以对任何人都不信任(18 世纪的英国谚语)。

安全专家经常会使用单词对，例如安全的/危险的、可信的/不可信的和无恶意的/有恶意的，而不是清楚区分它们。这些区分很重要，因为它们经常会被误解。它们的主要区别在于：

如果一个特定程序集不造成危害(例如修改磁盘、窃取数据和拒绝服务等)，那么它就是安全的。程序集的安全性或危害性实际上是个技术问题：该程序集想干什么？例如，它读取磁盘吗？那它可能会窃取秘密。它创建对话框吗？那它可能会创建许多对话框，使得无法使用机器。

指派给程序集的信任级别实际上是基于证据所授予的一组权限。例如，如果您认为来自 Internet 的程序集被授予了访问您的磁盘的权限，那么它可能会利用这一能力对您造成危害，因此不该授予这一权限。管理员通过适当配置他们的策略来表示他们的信任意见。“不要让来自 Internet 的不可信代码访问磁盘”是这样一个策略的示例。

一个程序集是恶意的还是无意的取决于程序集设计者的目的。能够创建文件的程序集可能是完全无恶意的，它也可能是创建文件来占用机器上的所有硬盘空间，构成一个拒绝服务的攻击。创建一个看上去像 Windows 口令对话框的程序集可能是想窃取您的口令。

遗憾的是，没有办法来检测编程人员的意图。如果有，那么安全性会很简单：使用不可思议的 ESP(第六感觉)来查看编程人员的意图，接着阻止有恶意的人编写的代码运行！在现实生活中，.NET 安全系统所能做的只是根据可用的证据限制程序集的能力。

注意，我们在技术上讲这些单词的方式和日常生活中一样。网球本来就是相当安全的，而斧头天生就是相当危险的。持有斧头的人有无恶意与斧头本身无关，但与他们的意图有关。是否信任他们做院子里的工作与您掌握的关于他们的可信度的证据有关。表示这种信任的策略可能就是“不要让带有斧头的陌生人进入院中”。

1.3.3 危害

现在，我们讨论促使人们攻击软件的原因，那就要看一下在攻击时会造成什么危害。在计算机安全中，我们需要考虑 3 种主要的危害类别。按照危害的严重性排列如下：

- 拒绝服务
- 未经授权访问数据
- 数据篡改

这些危害是极为广泛的，每一种都可以造成相当大的危害范围。它们会以某种方式来危害用户。

1. 拒绝服务

如果攻击者想阻止合法的用户做什么事，那么他们会采用拒绝服务(DoS)攻击。用于拒绝服务的技术有多种。例如，简单的 DoS 攻击可能会给 Web 服务器提供有害的输入，从而占用大量资源例如内存或磁盘空间。成功的针对服务器的 DoS 攻击会占用很多的内存，阻止它对合法用户的请求提供服务。

DoS 攻击经常使用这种方式占用所有有限的资源，这些资源可能是内存、磁盘空间、处理器周期、打印纸、文件句柄、网络带宽或是屏幕的占有权。最后一项常见于 World Wide Web 上：人们在访问 Web 页面时可能会有这样的经历，经常产生新的全屏幕的窗口广告。Internet Explorer 的早期版本允许按页面设计者的想法多次显示页面。

遗憾的是，尽管.NET Framework 对磁盘空间和打印纸提供了一些保护措施，对客户机的这些类型的 DoS 攻击实际是不可能停止的。所幸的是，它们通常仅是有些烦人，通过删除这些讨厌的进程或是重启机器就可以得到处理。然而要注意，如果被攻击的客户机正被调度程序使用，那么对于普通用户来说，只是很烦人的事可能就会产生非常严重的问题。

更为严重的是对服务器机器的 DoS 攻击。如果攻击者使得服务器停止向用户提供数据，那么给人带来的就不仅是烦恼了。如果那些用户要求及时访问数据进行工作，那么代价就很大了。譬如考虑一下延迟股票报价或其他定价数据的损失。有好几种用于对服务器进行 DoS 攻击的技术。不过主要有两种形式：攻击者提供导致服务器不能正常工作的有害数据，或是用大量假的请求来充斥服务器，从而严重减缓服务器的响应速度。

对服务器的 DoS 攻击通常都是分布式攻击(distributed attacks，DDoS)。在 DDoS 中，攻击者使得许多机器同时攻击服务器，使攻击效果成倍增强。通常，用于攻击的机器已经被攻击者攻击过了，无需授权就可使用它们。

著名的“smurf”攻击就是个很好的例子。现实生活中就有类似的恶作剧，爱开玩笑的人用您的名字填写许多“bill me later”杂志订阅卡，接着烦人的邮件大量涌向您。在 smurf 攻击中，攻击者发送了许多“请告诉我您的机器是否在 Internet 上”的请求给许多机器，但将发送者的地址伪装成您的服务器发送请求。当成千上万的收件人都试图告诉目标机器时，目标机器会花费所有时间来处理讨厌的通信，而不是服务于合法的客户机。

要获得有关 smurf 攻击的更详细内容，可以查看 http://www.pentics.net/denial-of-service/white-papers/smurf.cgi。

简单的 DoS 可以很容易实现，但却很难防止。通常，用于减轻对客户机的 DoS 攻击的启发式方法有些武断；例如，Internet Explorer 允许 Web 页面创建无限多个消息框，但并不允许在可见屏幕区域的外部创建它们或是占据整个屏幕。

在接下来的章节中，我们将讨论一些可能的攻击和防护措施，但是将着重介绍另外两种危害。用于阻止对服务器的 DDoS 攻击的技术超出了本书的范围，不过我们还将总体讨论一下使得服务器端代码足以抵御攻击的技术。

2. 未经授权的数据访问

坦率地说，攻击者对他们获取的任何数据都感兴趣。每台机器上都存储了大量不同的数据。我们可以将这些“数据”进一步分成三类数据：个人数据、配置数据和授权数据。

个人数据由存储在磁盘中的所有信息组成，例如 Word 文档、数据库或源代码。您的文件可能包含机密的信息。显然，如果攻击者可以读取这些文件，那么就会对您造成危害。如果数据落入未经授权的人手中，那么不管是无关紧要的(例如获取您的电子邮件地址的垃圾信息发布者)，还是会造成灾难性后果的(例如窃取秘密的源代码或机密的财务数据的对手)，都会造成潜在的危害。

攻击者也会寻找像个人收入状况、信用报告、病历和其他这类个人隐私数据。个人数据窃取是个正愈演愈烈的问题。现在许多国家都有法律规定，公司必须采取适当的措施来保护这类敏感数据。

配置数据由如何配置机器的信息组成。已安装了哪个版本的 IIS？哪个版本的 Office？默认的电子邮件客户是什么？等等。通常窃取配置数据不是为了它本身；而是为了将其用作更广范围的攻击的一部分。例如，攻击者希望知道网络上还有哪些机器没有用最新的安全补丁更新过！如果攻击者知道特定软件的一些弱点，那么获取配置数据的攻击可以更容易地实现这一点。

这并不是说攻击者没有这些信息就不会进行特定的攻击；额外的配置信息使得攻击

更为容易。具备可用的配置信息更像拥有一个包括周围那些丢三落四的人的名单；要闯入汽车中不需要这些信息，但有了它一切会更容易些！

最为危险的是窃取秘密身份的验证和授权数据。总有一些数据用于验证用户的身份，从而授予他们不同的权限。信用卡号、用户名和口令、密钥等都是用于验证用户身份的数据，从而授予不同的动作，例如进行购买、访问文件和解码秘密文档。

显然，总有更重要和不太重要的数据，而更重要的数据需要更小心地保护。泄露不太重要的数据也是不可原谅的。攻击者对机器上的几乎所有数据(例如用户名、安装的应用程序和服务以及环境变量值等)都感兴趣，因为它们会为下次如何发起更强有力的攻击提供一些线索。

3. 数据篡改

允许未经授权的人读取个人数据、配置数据和授权数据是很有害的。而允许他们改变数据就更糟糕了。

篡改个人数据的攻击者可以造成很明显的破坏(删除文件)，也可以造成不易察觉的细微改变(调换电子数据表中的一些重要数字)，等发现时为时已晚。

事实上，细微的攻击比明显的攻击更危险。至少如果您知道文件被删除了，就可以进行灾难修复(您确实有个恢复方案，对吗？)。可以通过最后的备份恢复文件。更为重要的是，您知道自己被攻击了。这意味着您可以发现弱点，修补它，追捕攻击者。然而，如果攻击很细微，那么就不知道自己被攻击，一切也就无济于事了。

如果机器配置泄密了，那么就很难预防将来的攻击。一旦攻击者通过在您的硬盘上安装他们自己的应用程序，或是用特洛伊木马取代您的应用程序来篡改数据，那么就很难阻止他们再次攻击您。狡猾的攻击者会用不易察觉的方法修改您的机器配置，这样以后可以更容易地进行进一步攻击。

通过一些攻击点控制您的机器之后，有准备的攻击者要做的第一件事就是安装一个“根工具包”。根工具包就是介绍攻击点的一些软件，攻击者在后面会利用它们。如果他们现有的入侵被检测到了并修复了他们利用的弱点，那么会有另一种入侵的方法。

如果授权和验证数据易被篡改，那么就很难阻止攻击者的再次攻击。如果攻击者可以创建新的账户、改变您的密码或是执行其他管理员任务，那么将来保护机器的安全会相当棘手。

攻击者通常使用一种特定形式的数据篡改作为攻击的一部分或是整个攻击，即repudiation(否认)。否认攻击就是用特定的数据来篡改数据，这种方法使得无法知道发起攻击的人。可以清除安全日志的攻击者比那些所有行为都被永久记录下来的人更难检测到。可以将其他一些人的姓名写入安全日志中的攻击者会对无辜的人造成相当大的伤害。

在成功攻击后清除系统日志只是否认攻击的一种。另一种更为危险的否认攻击就是通过精心的删除来实际造成危害；例如，假定可以从α账户向β账户转账，接着从α