# 安全的范围

安全。IT世界里一个令人难以置信的超载词。在不同的上下文中，它意味着很多不同的东西，但最终，它都是关于保护敏感和有价值的资源免受恶意使用。在它中，我们有许多基础架构和代码层，它们可能会受到恶意攻击，而且我们可以说应该确保所有这些层都得到适当级别的保护。当然，随着互联网的发展，以及越来越多的人使用我们的应用程序，网络犯罪分子试图以非法的方式访问这些应用程序的大门越来越大。此外，我们并没有经常采取适当的措施，以确保向公众提供一套有适当保障的服务。有时，即使采取了良好的防范措施，一些黑客仍然足够聪明，能够克服表面上看起来足够的安全障碍。IT基础设施中的三个主要安全层是网络、操作系统和应用程序本身。

## 网络安全层

这一层可能是IT界最熟悉的一层。

当人们谈论IT安全时，他们通常会想到网络级别的安全——特别是使用防火墙的安全。

尽管人们经常将安全性与网络级别联系在一起，但这只是针对攻击者的一个非常有限的保护层。

一般来说，它仅能保护IP地址和过滤网络包在网络中特定机器的特定端口。

在绝大多数情况下，这显然是不够的，因为这个级别的流量通常允许进入您的各种公开服务的公开开放端口，而不受任何限制。

在这些开放服务上可以针对不同的攻击，因为攻击者可以执行可能危及您的安全约束的任意命令。

有一些工具，比如流行的nmap(http://nmap.org/)，可以用来扫描机器以找到打开的端口。

在准备攻击时，使用这样的工具是很容易的第一步，因为众所周知的攻击如果没有得到适当的保护，就可以对这样的开放端口进行攻击。

一个非常重要的网络层安全的一部分,在web应用程序中,使用安全SocketsLayer(SSL)沿着线编码所有敏感信息发送,但这是相关的网络协议在应用程序级别比防火墙的网络物理层操作。

## 操作系统层

这一层可能是整个安全模式中最重要的一层，因为如果某个特定的应用程序受到危害，一个安全的操作系统(OS)环境至少可以防止整个主机机器崩溃。

如果攻击者以某种方式被允许对操作系统进行不安全访问，他基本上可以做任何他想做的事情——从传播病毒到窃取密码，或者删除整个服务器的数据，使其不可用。

更糟糕的是，他可能会在你不注意的情况下控制你的电脑，并利用它作为僵尸网络的一部分进行其他恶意行为。

我们可以在这个层中包括应用程序的部署模型，因为您需要知道您的操作系统的权限方案，以确保您的应用程序不会对您的机器授予不必要的特权。

应用程序应该尽可能地与主机的其他组件隔离运行。

## 应用层

本书的主要焦点将集中在这一层。

应用程序安全层指的是我们在应用程序中建立的所有约束，以确保在处理应用程序时，只有正确的人可以做正确的事情。

默认情况下，应用程序会受到无数攻击。

安全保护不当的应用程序允许攻击者从应用程序窃取信息、冒充其他用户、执行受限操作、破坏数据、访问操作系统级别以及执行许多其他恶意行为。

在本书中，我们将讨论应用程序级安全，这是Spring安全的领域。

应用程序级安全是通过实现几种技术来实现的，有一些概念可以帮助您更好地理解本书的其余部分。

这些是Spring安全解决方案的主要关注点，它可以为您的应用程序提供全面的安全防护。

在以下三个小节中，我将介绍

•验证

•授权

•acl

### 身份验证

身份验证过程允许应用程序验证特定用户是她所声称的用户。

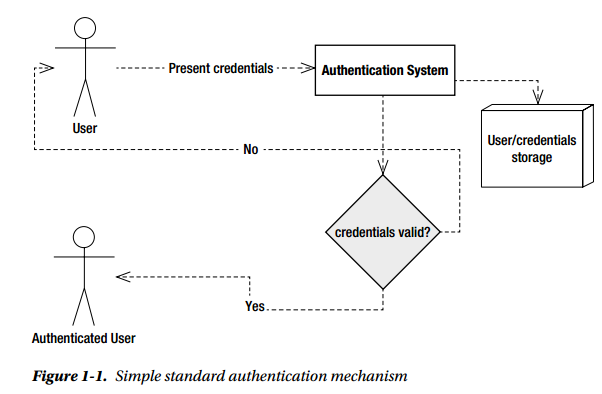
在身份验证过程中，用户向应用程序显示自己的信息(通常是用户名和密码)，其他人不知道这些信息。

应用程序获取这些信息，并试图将其与存储在数据库或LDAP1(轻量级目录访问协议)服务器中的信息进行匹配。

如果用户输入的信息与身份验证服务器中的记录匹配，则称该用户在系统中成功地进行了身份验证。

应用程序通常会在系统中创建一个表示这个经过身份验证的用户的内部抽象。

图1-1显示了身份验证机制。



### 授权

当对用户进行身份验证时，这只意味着系统知道该用户并已被系统识别。

这并不意味着用户可以在系统中自由地做任何她想做的事情。

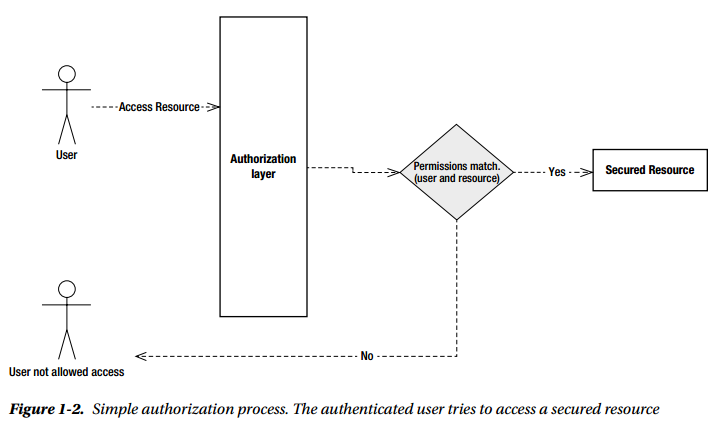
保护应用程序的下一个逻辑步骤是确定允许用户执行哪些操作，以及她可以访问哪些资源，并确保如果用户没有适当的权限，她就不能执行该特定操作。

这是授权过程的工作。

在最常见的情况下，授权过程将用户的权限集与在应用程序中执行特定操作所需的权限进行比较，如果找到匹配，则授予访问权限。

另一方面，如果没有找到匹配，则拒绝访问。

图1-2显示了授权机制。



### ACLs

访问控制列表(acl)是前一节中介绍的授权过程的一部分。

关键的区别在于，acl通常在应用程序的细粒度级别上工作。

acl只是资源、用户和权限之间映射的集合。

使用acl，您可以建立规则，如“用户John在博客帖子X上有管理权限”或“用户Luis在博客帖子X上有阅读权限”。

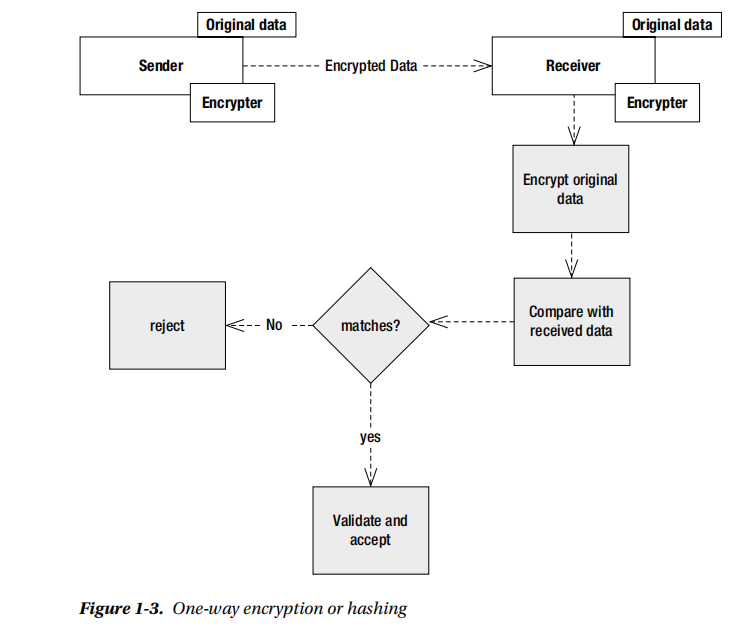
您可以看到三个元素:用户、权限和资源。

图1-2显示了acl是如何工作的，因为它们只是一般授权过程的一个特例。

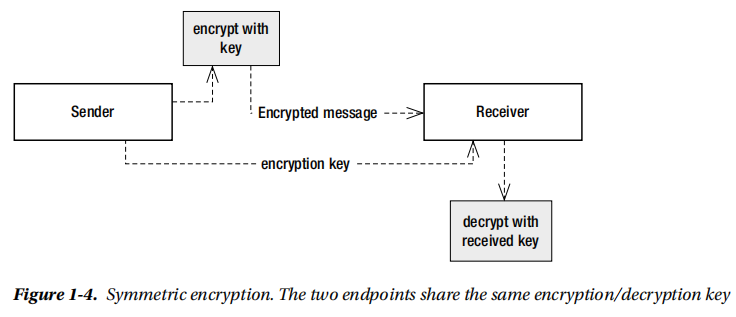
## 身份验证和授权:一般概念

在本节中，我将介绍并解释一些基本的安全概念，您将在本书的其余部分经常遇到这些概念:

* 用户：保护系统免受恶意攻击的第一步是识别合法用户并允许单独访问他们。用户抽象在系统中创建，并给出它们自己的标识。这些是以后允许使用系统的用户。
* 凭据：凭据凭据是用户证明自己的方式。通常，以密码的形式(证书也是表示凭据的一种常见方式)，它们是只有证书所有者知道的数据。
* 角色：在应用程序安全上下文中，角色可以看作是用户的逻辑分组。这种逻辑分组通常是这样做的:被分组的用户在应用程序中共享一组访问特定资源的权限。例如，所有具有“admin”角色的用户都将具有对相同资源的相同访问和权限。角色只是一种将权限分组以执行确定动作的方法，使具有这些角色的用户继承这些权限。
* 资源：这里的资源指的是，在这个上下文中，我们想要访问的应用程序的任何部分，以及需要对未经授权的访问进行适当保护的部分——例如，URL、业务方法或特定的业务对象。
* 权限：权限指访问特定资源所需的访问级别。例如，可能允许两个用户读取特定的文档，但只允许其中一个用户写入。权限可以应用于单个用户，也可以应用于共享特定角色的用户。
* 加密：这允许您加密敏感信息(通常是密码，但也可以是其他信息，比如cookie)，以便即使攻击者访问加密版本，也无法理解这些信息。其思想是，您永远不会存储密码的纯文本版本，而是存储一个加密的版本，以便只有密码所有者知道原始密码。加密算法主要有三种:
  + 单向加密：这些算法，称为哈希算法，取一个输入字符串并生成一个称为消息摘要的输出数字。这个输出数字不能转换回原来的字符串。这就是为什么这项技术被称为单向加密。下面是使用它的方法:请求客户机对字符串加密并将加密的字符串发送到服务器。例如，服务器可以访问以前注册过程中的原始信息，如果访问了，它可以对其应用相同的哈希函数。然后，它将这个散列的输出与客户机发送的值进行比较。如果匹配，服务器将验证信息。图1-3显示了这个方案。通常，服务器甚至不需要原始数据。它可以简单地存储散列版本，然后将其与来自客户机的传入散列进行比较。



* + 对称加密：这些算法提供了两个功能:加密和解密。文本字符串被转换为加密形式，然后可以转换回原来的字符串。在这种方案中，发送方和接收方共享相同的密钥，以便能够对通信两端的消息进行加密和解密。这个方案的一个问题是如何在通信的端点之间共享密钥。一种常见的方法是使用并行安全通道发送密钥。图1-4显示了工作中的对称加密。



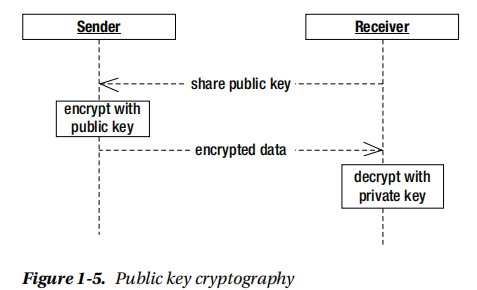
* + 非对称加密：在这个方案中，加密使用的密钥与解密使用的密钥不同。这两个密钥称为公钥(用于加密消息)和私钥(用于解密消息)。与对称加密相比，这种方法的优点是不需要共享解密密钥，因此只有信息的预期接收者才能解密消息。通常情况是这样的:

•消息的目标接收者与所有感兴趣的人共享其公钥发送信息。

•发送方使用接收方的公钥加密信息，并发送消息。

•接收者使用其私钥解密消息。

•没有其他人能够解密消息，因为他们没有接收方的私钥。



加密的使用实现了另外两个安全目标:

•机密性：属于一个用户或一组用户的潜在敏感信息应该仅对该用户或组可访问。加密算法是实现这一目标的主要助手。

•完整性：由有效用户发送的数据不应被第三个实体在发送到服务器的途中或在其存储中更改。这通常是通过使用单向加密算法来实现的，这种算法几乎不可能改变输入并生成加密后的消息，其加密哈希与原始消息相同(从而欺骗接收方，使其认为它是有效的)。

## 需要保护的

并不是应用程序的每个部分都需要强大的安全模型，甚至不需要任何安全性。

例如，如果您的应用程序的一部分应该为所有感兴趣的人提供静态内容，那么您可以简单地提供这个内容。这里可能没有需要处理的安全问题。无论如何，在开始处理新应用程序时，您应该考虑应用程序将具有的安全约束。您应该考虑以下列表中的问题，以及它们是否适用于您的特定用例:

* 身份管理：很有可能，您的应用程序需要确定将要使用它的不同用户的身份。通常，您的应用程序会为不同的用户做不同的事情，因此您需要一种将用户与特定功能关联起来的方法。您还需要确保保护每个用户的身份信息，使其不会受到危害。
* 安全连接：在Internet环境中，任何人都可能访问您的系统并窃听其他用户访问您的系统，您很可能希望使用某种传输层安全(例如SSL)来保护敏感数据的通信。
* 敏感数据保护：需要保护敏感数据不受恶意攻击。这既适用于通信层，也适用于单个消息传输，也适用于凭证数据存储。加密将在不同的层中使用，以实现最安全的应用程序。

## 更多的安全问题

到目前为止，有更多的安全问题需要解决。因为这是一本Spring安全书，而不是一本通用的应用程序安全书，所以我们将只讨论与Spring安全相关的内容。但是，我认为重要的是您要理解，与Spring security直接处理的安全问题相比，还有更多的安全问题。下面是一些最常见问题的简要概述。这只是为了让您知道它们的存在，我建议您咨询另一个来源(比如一般的软件安全教科书)，以便更好地理解所有这些关注点:

* SQL(和其他代码)注入：验证用户输入是应用程序安全性的重要组成部分。如果数据没有经过验证，攻击者可能会将任何类型的字符串写入输入(包括SQL或服务器端代码)并将该信息发送到服务器。如果没有正确编写服务器代码，攻击者可能会造成严重破坏，因为她可以在服务器上执行任意代码。
* 拒绝服务攻击：这些攻击包括使目标系统对目标用户不响应。这通常是通过让服务器充满请求来完成的，这样它就可以利用服务器的所有资源并使其对合法请求不响应。
* 跨站点脚本编写和输出环境卫生：一种注入可以在目标是应用程序的客户端部分的地方进行。攻击者可以让应用程序在返回的web页面中返回恶意代码，从而在用户的浏览器中执行代码。通过这种方式，攻击者将使用真实用户的经过身份验证的会话以不可见的方式执行操作。

## Java安全选项

Java和Java EE开箱即用的安全解决方案非常全面。它们涵盖了从低级权限系统、通过加密api到身份验证和授权模式的各个领域。Java提供的安全api列表非常广泛，如下所示:

* Java加密体系结构(JCA)：这个API提供了对加密算法的支持，包括哈希摘要和数字签名支持。
* Java加密扩展(JCE)：该API主要提供对字符串进行加密和解密的功能，以及对称算法的密钥生成功能。
* Java认证路径API (CertPath)：此API提供了将数字证书的验证和验证集成到应用程序中的综合功能。
* Java安全套接字扩展(JSSE)：这个API提供了一组标准化的特性，以在Java中提供对SSL和TLS协议(客户机和服务器)的支持。
* Java身份验证和授权服务(JAAS)：此API为Java应用程序中的身份验证和授权提供服务。它提供了一个可插入的系统，身份验证机制可以独立地插入到应用程序中。

Spring安全的主要关注点是身份验证/授权领域。因此，它主要与JAAS Java API重叠，尽管它们可以一起使用，您将在本书后面看到。大多数其他api都在Spring Security中使用。例如，在X509AuthenticationFilter中使用CertPath，在spring-security-crypto模块中使用JCE。

## 总结

在本章中，我从一般的角度介绍了安全性。我以非常抽象的方式解释了IT安全中的主要关注点，特别是从应用程序的角度。我还简要介绍了支持不同级别安全性的主要Java api。

您可以看到，这一章是对安全问题的快速概述。超出了本书的范围，不能进一步讨论一般主题，尽管我们将在将其中一些应用到Spring安全性时对其进行更深入的研究。显然，这一点都不像一个全面的软件安全指南，如果您有兴趣了解更多关于软件安全的一般知识，您应该查阅专门的文献。下一章将介绍Spring安全性。

# 引入Spring Security

在本章中，您将了解什么是Spring安全性，以及如何使用它来解决应用程序的安全问题。

我们将构建一个具有Spring安全性的简单应用程序。

我们将从一个没有任何安全性的基于servlet的web应用程序开始，然后我们将以声明性的、非侵入性的方式向其添加安全性。

在本章中，我们还将查看框架的源代码、如何构建它以及组成强大的Spring安全项目的不同模块。

## Spring Security是什么?

Spring Security是一个框架，致力于以开发人员友好和灵活的方式为Java应用程序提供完整的安全服务。它遵循Spring框架引入的成熟实践。Spring Security尝试处理应用程序中的所有安全层。此外，它还提供了大量的配置选项，使其非常灵活和强大。回想一下第一章的介绍，可以这样说，Spring安全仅仅是一个构建在Spring框架之上的全面的身份验证/授权框架。虽然大多数使用该框架的应用程序都是基于web的，但Spring Security的核心也可以在独立的应用程序中使用。许多事情使Spring安全立即吸引了Java开发人员。举几个例子，我编制了以下列表:

* 它构建在成功的Spring框架之上：这是弹簧安全的一个重要优势。Spring框架已经成为构建企业Java应用程序的“途径”，这是有充分理由的。它是基于良好的实践和两个简单而强大的概念构建的:依赖注入(DI)和面向方面编程(AOP)。同样重要的是，许多开发人员都有使用Spring的经验，因此他们可以在项目中引入Spring安全性时利用这种经验。
* 它为许多身份验证模型提供开箱即用的支持：更重要的是，Spring安全支持开箱即用的集成，包括轻量级目录访问协议(LDAP)、OpenID、表单身份验证、证书X.509身份验证、数据库身份验证、Jasypt加密等等。所有这些支持都意味着Spring安全可以适应您的安全需求——不仅如此，如果您的需求发生了变化，Spring安全还可以进行更改，而开发人员无需付出太多努力。从业务的角度来看，这也是很重要的，因为应用程序既可以适应公司身份验证服务，也可以实现自己的身份验证服务，因此只需要简单的配置更改。这也意味着可供您编写的软件要少得多，因为您正在使用大量由大型活跃用户社区编写和测试的现成代码。在一定程度上，您可以相信这段代码能够正常工作，并有信心地使用它。如果它不工作，您可以随时修复它，并向负责维护项目的人员发送补丁。