Java 鉴别和许可服务机制的研究

The Research of the Mechanism of Java Authentication and Authorization Service

舒 昶 吕述望

(中国科学院研究生院信息安全国家重点实验室 北京 100039)

Abstract Java is a Net-Oriented computing technology. When it is applied to a distributed system, we must think over the security of the distributed system. If the system contains some parts that are potentially hostile to each other, the service login and location must be managed reliably. Java Authentication and Authorization Service is a standard API designed for authenticating user and allocating authority. By JAAS and JDK1.2, an application can provide codesource-based, user-based access controls or the combination of the two. This paper expatiates the mechanism of JAAS based on the Kerberos protocol.

Keywords Java Authentication and Authorization Service, Kerberos protocol, Authentication, Access control, Subject, Principal, Credential, Callback

1. 引言

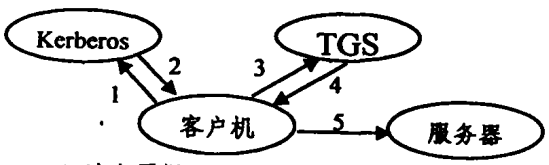
JDK1.2运用安全策略来决定对运行代码赋予访问许可权,而且这种决定依赖于代码的特性,例如代码从何而来,是否被数字签署以及被谁签署。这样一种代码中心型的访问控制并不常见。在成熟的操作系统中最常见的传统安全度量,是用户中心型的,它们以谁运行程序为基础申请控制,而不是以哪个程序被运行为基础。但是代码中心的访问控制被大大看好,因为 Internet 用户基本上保持着不变的身份而且运行可执行内容。另一方面,用户可能更信任某一动态代码,而且希望以更多的特权来运行这段代码。因此以代码中心方式来控制动态代码的安全性是自然的。Java 正越来越广泛地用于多用户环境中,要和不同的用户打交道,因此必须根据用户的身份赋予不同的许可权。JAAS 正是为鉴别用户和分配许可权而设计的一套标准程序界面。JAAS 具有可延展性、可插入性和兼容性。目前现存的标准,如:普通安全服务应用程序界面(GSS-API)和简单鉴别与安全层应用程序界面(SASL),都支持鉴别。SASL 构建了为基于连接的协议而提供鉴别支持的框架,因此适合用于执行网络鉴别的应用程序。如同 JAAS 一样,SASL 也是模块化构造。GSS 进程,可以在 SASL 框架下插入。而 JAAS 也对独立非连接性环境下的鉴别提供支持。因此 JAAS 和 SASL/GSS 互为补充,为鉴别提供本地的和基于网络的支持。

2. Kerberos 协议及其引用

2.1 Kerberos 协议

Kerberos 是为 TCP/IP 网络设计的可信第三方鉴别协议。网络上的 Kerberos 服务起着可信仲裁者的作用,它是一种网络安全协议,可以为 JAAS 和 SASL/GSS 的共存结构提供鉴别的环境。为便于问题的说明,简单介绍一下 Kerberos 工作原理(见图1)。

客户首先从 Kerberos 请求一张票据作为票据许可服务(Ticker-Granting Service,TGS),该票据利用用户的秘密密钥加密后发送给用户。为了使用特定的服务器,客户需要从 TGS 中请求一张票据。TGS 将票据发回给客户,客户将此票据呈示给服务器和鉴别器,如果客户的身份没有问题,服务器便会让客户访问该服务。



1. 请求票据-许可票据 2. 票据-许可票据
3. 请求服务器票据 4. 服务器票据 5. 请求服务

图1

2.2 Kerberos 协议的引用

当最初登录到 Kerberos(为获得用户的 Kerberos Ticket Granting 标签)和底层的操作系统时,登录应用程序可以插入到 JAAS 登录模块中来鉴别用户。通过安装一个 Kerberos 登录模块,用户可以不必再执行额外的步骤,例如:执行 Kinit 命令得到标签。当用户执行客户端应用程序,以通过网络到达某些使用 Kerbero 协议的服务器来进行鉴别的时候,这些应用程序可以使用 SASL,它基本上可以插入合适 Kerbero 机制来执行真正的鉴别。

3. JAAS 的鉴别机制

3.1 主题与主体

用户常常依赖于计算服务的辅助来执行工作,而计算服务通常依赖于用户名来鉴别它的用户,然而,用户对每一个服务所用的名字可能不尽相同,事实上可能对每一个服务都有不同的名字。此外,一个服务也可能是其他服务的用户。因此可能有多重的名字。JAAS 用主题(subject)来代表计算服务的所有用户。因此,用户和计算服务都代表了主题。同样,用主体(principal)代表与一个主题有关的名字。主题可能有多个名字(一般对每个相互作用的服务都有一个名字),基本上一个主题包含一组主体。

只有当主题成功地被一个服务鉴别时,主体才与某一个

舒 昶 硕士生,研究方向:信号与信息处理、信息安全。吕述望 教授,博导,研究方向:密码学与信息安全。

主题相联系。鉴别体现了一方鉴别另一方的身份的过程，而且必须在安全方式下执行。一般情况下，鉴别要求主题出示一定形式的证据来证明它的身份。这样的证据也许只是主题希望知道的或者已有的信息。当主题试图鉴别于服务时，它通常在提供它的身份证据的同时也给出它的名字。如果鉴别成功，服务程序将会根据所给定的名字为主题分配一个特定的服务主体。这样，应用程序和服务可以仅索引与这个主题相关的主体来判定主题的身份。

3.2 证件

除了主体信息，一些服务也许希望将主题和其他与安全有关的特性和数据相联系。JAAS 把这样的与安全有关的一般特性叫做证件(credential)。证件中可能包含将来其他服务鉴别主题的信息。Kerberos 标签就是这样的一个证件。证件也可能只包括或索引一些仅仅使主题执行某一活动的数据。例如密钥，就是一种可以使主题签名或加密数据的证件。

3.3 可插入和可堆叠的鉴别

传统的服务程序要求一个主题提供它的名字或者某种形式的证据来鉴别。证据的类型取决于特定服务的安全参数。JAAS 鉴别支持这样的一种结构，即允许系统管理员插入正确的鉴别服务来满足安全需要。这种结构也使应用程序能够与底层的鉴别程序保持独立。因此，当新的鉴别服务程序出现或者当目前的服务过期时，系统管理员可以轻易地将它们插入，而不用修改或者重新编译现有的应用程序。

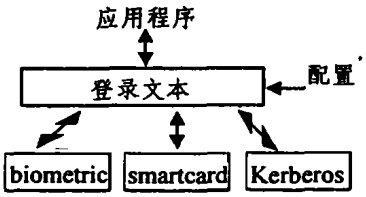


图2 可插入的鉴别

当试图鉴别一个主题时，一个应用程序调入鉴别框架，在 JAAS 中该框架被定义为一个登录文本(login context)。这个登录文本将参考那些决定鉴别服务或者在这个应用程序中被插入的登录模块(login module)的配置。因为应用程序只和登录文本界面，所以它完全与配置过的登录模块相独立。

每个登录模块都以不同的方式来鉴别主题。例如，传统的基于口令的登录模块提示输入用户并验证密码；智能卡登录模块告知主题将它的卡插入读卡器并验证 PIN；生物型的登录模块提示输入用户名并且验证主题的指纹。针对应用程序的安全要求，系统管理员可以配置正确的登录模块。事实上，系统管理员可以在一个应用程序中插入多个登录模块(见图3)。

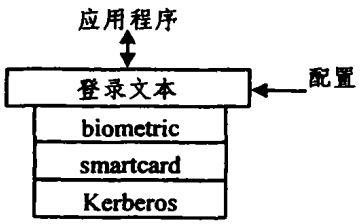


图3 可堆叠的鉴别

一个主题按照配置中规定的顺序被登录模块鉴别。一般地，不管前一个登录模块是否登录成功，主题继续被堆栈里的登录模块鉴别。只有当所有必需的登录模块(由配置决定)登录成功之后。登录文本才会返回一个鉴别成功标志给调用程序。为保证这一点，登录文本用两个阶段执行这个鉴别步骤。两个阶段必须成功地完成，登录文本才能返回标志总体鉴别成功的状态。

(1)登录文本调用一个配置过的登录模块，而且指引它验证主题的身份。如果所有必需的登录模块成功地通过这个阶段，登录文本就进入第二个阶段。

(2)登录文本再次调用每一个配置过的登录模块，指引它正式的执行鉴别过程。在这个阶段中，每一个登录模块都会将与主题和任何相关的含有用户名和主题的证件的主体联系起来。

因此一旦总体鉴别过程完成，调用程序可以遍历一个主题的主体集合来得到它的各种身份，而且可以遍历一个主题的证件来访问补充数据。一些登录模块可能只和主题的证件(而不是主体)相联系。例如，智能卡模块可以通过验证被提供的 PIN 来鉴别主题，如果鉴别成功，它只将主体与索引到卡上的密钥的证件相联系。在这种情况下，智能卡模块就不将主题与主体相联系。

不管哪一个阶段失败了，登录文本都会调用每一个配置过的登录模块，且指示它们放弃所有的登录尝试。这时每一个登录模块都清除掉它所有的已与鉴别尝试相联系的相关状态。在这个鉴别过程中，登录模块有机会和能力共享信息，不管它是不是依赖于安全要求。共享信息的一个原因是为了达到单次登录的目的。

下面讨论一下关于配置的问题：JAAS 登录配置规定登录模块在一定的应用程序下被插入。配置语法基于 PAM 的定义。见下例。

```
Login{
   com. sun. security. auth. sampleLoginModule REQUIRED debug=
        true;
   com. sun. security. auth. SolarisLoginModule REQUIRED;
}
```

配置中每个项目通过一个应用程序名字被索引，如此例中的 Login，而且包含一系列为这个应用程序配置的登录模块。鉴别程序按照列出的顺序在这个队列中运行，而 REQUIRED 这个标志值控制着总体行为。以下是有效标志值的描述：

·REQIRED：此登录模块必须成功。然而，不管它是否成功，鉴别仍按照登录模块队列里的顺序继续进行。

·REQUISITE：此登录必须成功。如果它成功了，鉴别仍按照登录模块队列中的顺序继续执行。如果失败了，控制权立刻返回应用程序(鉴别不继续执行)。

·SUFFICIENT：此登录不要求必须成功。如果它成功了，控制权立刻返回应用程序(鉴别不继续执行)。如果失败了，鉴别按照登录模块队列中的顺序继续执行。

·OPTIONAL：此登录模块不要求必须成功。不论成功或者失败，鉴别仍然按照登录模块队列中的顺序继续执行。

只有当所有的 REQUIRED 和 REQUISITE 登录模块成功时，总的鉴别过程才通过。如果对一个应用程序没有 REQUIRED 和 REQUISITE 登录模块被配置，那么至少一个 SUFFICIENT 或 OPTIONAL 登录模块必须通过。也可以用 LoginModule 来定义选项以支持调试/检测能力。

3.4 回调(callback)

通过使用登录文本，应用程序和底层的登录模块保持独

立。然而，登录模块可能需要在任何形式的应用程序中插入。不管登录模块插入到何种应用程序之下，它必须能够通过呼叫程序收集信息和对主体显示信息。例如，一个需要用户名和口令信息的登录模块要有提示主体输入这些信息的能力，而不需要知道是否调用程序有 CGI。

登录文本通过允许应用程序规定一个回调(callback)来解决这个问题，通过底层的登录模块可以与主题相互作用。应用程序提供一个回调工具给登录文本，这样登录文本可以直接把它传给每个登录模块。所以登录模块可以调用回调来收集或者显示相关信息。由应用程序实现的回调，本身就知道是创建一个图像窗口还是简单地使用一个标准输出流。回调的设计和使用见图4。

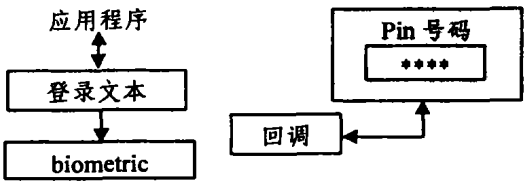


图4　回调的设计和使用

4. 访问控制

JDK1.2 SecurityManager 类实现基于源代码的访问控制，一种基于代码从哪里来和谁运行这些代码的控制方式，从而提高了安全管理。只有通过基于源代码的核查和基于主题的核查，JAAS SecurityManager 才能对敏感的资源进行访问。

4.1　许可权的赋予

鉴别是许可的基础，尤其是一旦应用程序知道了主题的准确身份，它就可以确切地规定主题可以执行哪些操作。因为主题只代表为用户保存相关信息的没有名字的容器，而主体代表这个主题被鉴别过的身份，所以赋予主题的一组许可权依赖于这个主题相联系的主体，而不是主题本身。也就是，许可权基于被鉴别的主体被赋予一个主题，被赋予的确切许可权由外部的访问控制策略来配置。

JDK1.2策略语句是为基于原码的访问控制规定的，JAAS 语句是它的一个简单延伸，并且加上了赋予许可权语句中的主体项目。因此基于源代码的策略对一个源代码赋予许可权(一个 URL 以及签名的别名)，然而 JAAS 策略对主体赋予许可权(以主体类名和主体本身的名字来识别)。

4.2　实施以用户为中心的安全策略

在 JAAS 中，应用程序将主题与一系列操作相联系，以期得到这个主题的访问控制。当执行操作时，JAAS SecurityManager 取得相关主题，并在允许它执行敏感操作和访问敏感资源之前核查它是否被赋予了必要的许可权。如果主题没有被赋予充足的许可权，SecurityManager 就启动 SecurityException。

运行时，某个操作可能要调用其他的操作，因此建立了一系列的嵌套操作。当这种情况发生时，每一个嵌套操作都被推入了一个堆栈。当 JAAS SecurityManager 必须做出访问控制决定时，它访问堆栈并找到所有与堆栈中操作有关的主题。它检查并确定堆栈中的每一个主题都被赋予了必要的许可权。如果一个或者更多的主题没有必要的许可权，SecurityManager 就启动 SecurityException。结果，对整个堆栈的总许可权实际上等于对堆栈中每一个主题所赋予的许可权的交集。

在某些情况下，一个操作和它的相关的主题可能希望行使它自己的许可权，而且，为了得到全面的访问，希望不依赖于早先在堆栈中有相同许可权的操作。这时情况发生在当一个正在执行的操作与它相关的主题，查询另一个与不同主题相关的操作，以完成自身的一定任务时，被查询的操作当然具有必要的许可权来执行所需任务，但一般来说早先的那个操作并没有。如果还是使用早先规定的访问控制模式，访问可能被拒绝。为解决这个问题，被查询的操作，如果有许可权，则可以定义自身享有特权。当享有特权时，该操作不需要它的调用程序或请求者被赋予和它一样的特权，就可以对被赋予许可权的资源进行全面的访问。只有操作本身以及它有关的主题，需要有必要的许可权。

如果一个享有特权的操作随后又调用一个非特权的操作，那么再一次地，访问操作权决定于有特权的和后来的无特权的操作这两者的主题的许可权的交集。

小结　Java 技术与操作平台无关，正是基于这一点，它特别适合于网络计算方面的应用，是为 Internet 和 Web 编写移动代码、可执行代码的最好工具。由于在网络应用方面，安全是必须考虑的问题，一个高层的完整的服务必须是可靠的，JAAS 是为处理分布式系统的安全而设计的应用程序界面，它提供了可插入或可堆叠的鉴别机制，实施以用户或代码为中心的访问控制策略。

参 考 文 献

1　Neuman B C, Ts'o T. Kerberos: An Authentication Service for Computer Networks. IEEE Communications, 1994, 32(9): 33～38
2　(美)Bruce Schneier 著. 应用密码学. 机械工业出版社
3　Arnold K, Gosling J. The Java Programming Language, Second Edition. Addison-Wesley, Reading, Mass., 1998
4　(美)Li Gong 著. Java2平台安全技术-结构、API 设计和实现. 机械工业出版社
5　Saltzer J H. Protection and the Control of Information Sharing in Multics. Communications of the ACM, 1974, 17(7): 388～402

更正："国际知识发现与数据挖掘工具评述"一文发表于我刊2001年第28卷第3期 P101～108，此文系翻译整理自 Goebel M. and Gruenwald L. 所发表 A survey of data mining and knowledge discovery software tools, SIGKDD Explorations, 1999, 1(1): 20-33一文。