基于移动代理的分布式防火墙系统

Distributed Firewall System Based on Mobile Agent

胡宝芳 （中华女子学院山东分院 计算机系 山东 济南 250000）

摘 要：针对现在 VPN 网络中的防火墙问题，结合 VPN 和移动代理这两大技术，提出了一种新型的适用于 VPN 网络的基于移动代理的分布式防火墙系统—VPNAgent 系统，在该系统中由网络防火墙派遣移动代理去客户机处检测和签名数据，并护送合法数据到网络服务器，这样数据包经过网络防火墙时只需由护送该数据包的护送 Agent—VPNClientAgent 出示签名以证明该数据包是合法的而不必解包，这种结构解决了传统边界防火墙不适合于 VPN 连接的状况，使现有的 VPN 网络连接解决方案更安全可靠。

关键词：移动代理 分布式防火墙 VPN 签名

1 引言

传统的边界式防火墙依赖于物理上的拓扑结构，它从物理上将网络划分为内部网络和外部网络，这一点影响了防火墙在 VPN 中的应用。根据 VPN 的概念，它对内部网络和外部网络的划分是基于逻辑上的，而逻辑上同处内部网络的主机可能在物理上分处内部和外部两个网络。对于这个问题的传统的边界防火墙是将远程‘内部’主机和外部主机的通信依然通过防火墙隔离来控制接入，而远程‘内部’主机和防火墙之间采用‘隧道’技术保证安全性，这种方法使原本可以直接通信的双方必须绕经防火墙，不仅效率低，而且增加了防火墙过滤规则设置的难度，因而传统的边界防火墙不适合于在 VPN 网络中使用。为此引入了分布式防火墙（Distributed Firewall）的概念。分布式防火墙是指物理上有多个防火墙实体在工作，但在逻辑上只有一个防火墙。从管理者角度来看，他不需要了解防火墙分布细节，只要清楚有哪些资源需要保护，以及资源的权限如何分配即可[1]。

移动代理是一段自主程序，可以在异构网络中按照自己的意愿从一台计算机迁移到另一台计算机[4]。也就是说这种程序可以选择何时迁移以及迁移的目的地，它能在任意点悬挂，把自己传送到另一台机器上恢复执行。它具有自主计算、平台无关性以及能够对环境变化做出感知和应变等优点，非常适合分布式防火墙系统对动态性和灵活性等方面的要求。

本文的目的就是介绍一种新型适用于 VPN 网络的的基于移动代理的分布式防火墙系统 – VPNAgent 系统，该系统中有一个移动代理 – VPNClientAgent，作为主机防火墙作用在客户端，检测数据包并对合法的数据包签名，然后护送数据包到服务器处。VPNAgent 系统中还有一个静态代理一 StaticAgent，其嵌入到传统的边界防火墙即网络防火墙上，用于检测数据包的签名，签名有效的数据包不用解包即可通过网络防火墙。这种系统把检测数据包的工作分配给了各客户机，属于典型的分布式防火墙系统。

本文分两部分介绍 VPNAgent 系统，一是其体系结构，二是系统实现。

2 VPNAgent 系统体系结构

假定有两个国家进行贸易交换，A 国从 B 国按一定的进口条令进口货物，为了保证货物符合条令，A 国可以先派遣一个客户代理到 B 国去预先检测货物，如果货物检测合格，就给货物颁发签证，表示这个货物符合输入条令，然后客户代理就带着具有签名的货物返回 A 国。当货物到达 A 国的边境时海关信任该签证，就无需再对货物进行检测。

由此可以考虑构建一个 VPNAgent 系统来实现信息的安全交换。在这个系统中有一个类似于客户代理的移动代理 – VPNClientAgent，如果客户端 B 想与在防火墙（边境）后面的服务器 A 交换信息，防火墙要先派

遣一个移动代理去客户 B 处检测信息是否合法,如果合法则给其加密,颁发签名,然后再由移动代理护送信息到 A 的防火墙处,防火墙上嵌有一个静态通信代理 – StaticAgent,它会验证签名是否有效,如果签名有效,就允许数据包进入内部网络。这一切都是在 VPNAgent 协议定义的规则下进行的。下面将介绍该系统的体系结构和其应用的协议。VPNAgent 系统包括 VPNClientAgent、StaticAgent 和 VPNAgent 协议。

2.1 VPNClientAgent

VPNClientAgent 是一个移动代理,能从防火墙(服务器)迁移到指定的客户处做一定的检测工作,这个代理包含代码和数据(特定检测目的代码和安全策略数据),它是和一个静态通信代理同时工作的。VPNClientAgent 作为防火墙的一个检测代表,在各客户端进行检测,它利用防火墙嵌入其上的安全规则进行数据包检测。

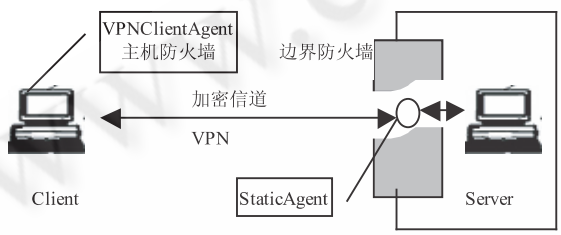

图 1　VPNAgent 系统的体系结构

VPNClientAgent 是一个特殊场合的用户化的防火墙,它能按客户的需要和当前连接的需要产生,不能用于普遍的或不同时间的其它场合。

VPNClientAgent 的任务主要是检测和签名,其结构包括检测单元、加密单元和签名单元。VPNClientAgent 和加密协议一起进行黑箱操作,代理检测信息,用一个默认的加密协议把信息分割成数据包,然后把数据包加密,传入签名单元。

图 2 描述了一个 VPNClientAgent 的工作流程,数据包依次经过检测、加密和签名过程。

2.2 StaticAgent

StaticAgent 是一个静态通信代理,是防火墙的看门人,它的职责是在防火墙上只为特定的合法数据包打开一个通道,它控制通过防火墙的数据包的路由过程。

StaticAgent 的职责可以被归纳为三点:

1. 验证绑定到数据包的通信 VPNClientAgent 的签名。

2. 确保 VPNClientAgent 起作用。

3. 将合法的数据包路由到目的地。

图 3 描述了 StaticAgent 的工作流程,StaticAgent 包括签名验证单元和路由单元,在签名验证单元中,StaticAgent 先确定收到的的数据包有无签名,丢弃无签名的数据包,并向用户发送丢弃报告,然后验证签名是否正确,如果正确则让其通过防火墙,再路由到目的主机,如果不正确则按照用户指定的安全策略对其进行处理。

StaticAgent 和 VPNClientAgent 的签名算法共享密钥,如果签名有效,StaticAgent 会将数据包的加密部分传到目的服务器,否则丢弃。

2.3 VPNAgent 协议

VPNAgent 系统中所用到的主要协议:

Identification verification and authentication protocol(身份证明与实体认证协议):一个或多个信息的参与者需要彼此验证身份。系统假定每个参与者都有一个可信任的 CA(certification authority,证书权力机构)颁发的 X.509 证书,其中表明用户的信息,包括他的验证公钥和交换密钥。当然,也可以用其他的证书。

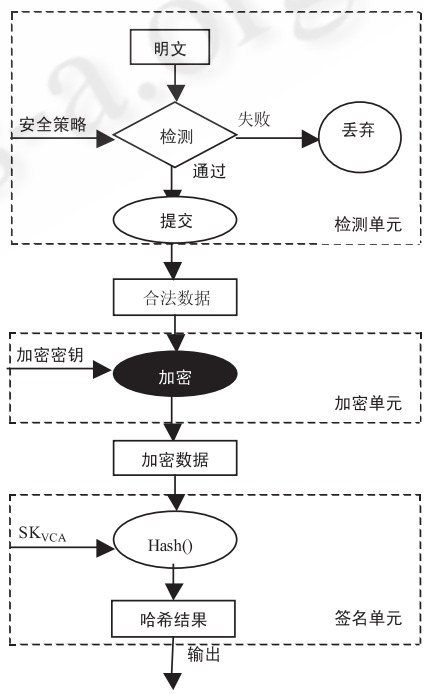

图 2　VPNClientAgent 的工作流程图

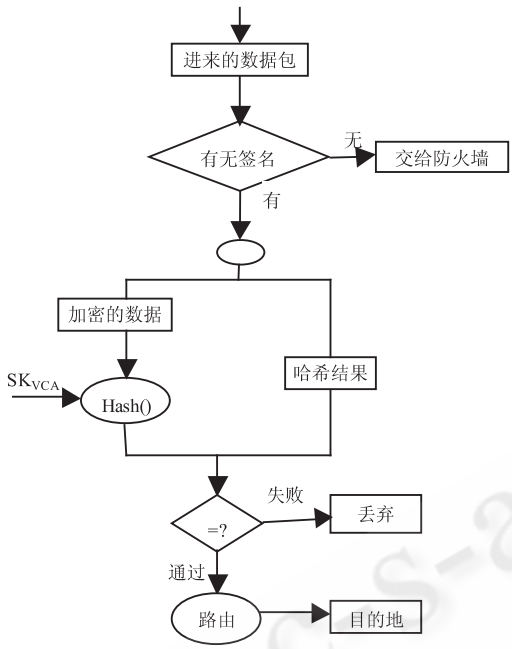

图 3　StaticAgent 工作流程图

Keys exchange protocol（密钥交换协议）：VPNAgent 协议用对称和非对称算法来交换密钥。VPNAgent 协议用一个非对称密钥交换结构作为验证方法的底层结构，并用初始化交换密钥来得到公钥。

Hash and signature algorithms（哈希签名算法）：加密和解密协议作用于 VPN 网络，VPNAgent 会用数字签名算法来加密 Agent 的签名。

3　系统实现

以某企业的网络系统为例来验证本系统的应用。

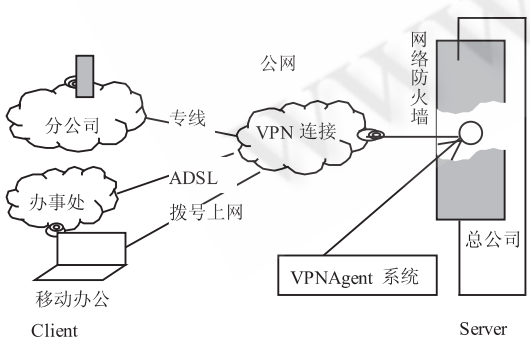

图 4　某企业网络拓扑结构

该企业的网络拓扑结构如图 4 所示，包括企业内部网络、分公司网络、办事处和移动办公人员，分公司、办事处和移动办公人员与总公司之间实现的是 VPN 连接，分公司与总公司之间是专线 VPN 连接，办事处是 ADSL（Asymmetrical Digital Subscriber Loop，非对称数字用户环线）上网，移动办公人员属拨号 VPN，分公司、办事处和移动办公人员属于客户。每个客户是可信的，有可信任的第三方颁发的数字 ID 证书，不会修改移动代理的代码。客户端要求有一个和本地网络的加密信道，客户用他的 ID 证书（如 X. 509 证书）和其它信息如公钥等来向防火墙提交验证，客户机允许移动代理在其机器上的运行。职员交流的信息通过防火墙传送，有不同的安全级别。

在内部网络服务器和客户机上都安装一个移动代理平台 – VPNClientAgent，其由 JAVA 实现，它完成对要发送到服务器的数据包的检测、加密和签名工作，并护送数据包到服务器处，交给服务器的防火墙。

本系统选择 Xfilter 个人防火墙来作为服务器的防火墙，它使用 Winsock 2 SPI 技术，工作在应用层。Xfilter 的核心功能有如下几部分：

1）根据应用程序访问规则可对应用程序连网动作进行过滤。

2）对应用程序访问规则具有自学习功能。

3）可实时监控、监视网络活动。

4）具有日志，以记录网络访问动作的详细信息。

5）被拦阻时能通过声音或闪烁图标给用户报警提示。

图 5　VPNClientAgent 工作界面

对于 Xfilter 的这些功能暂时只选择功能 1,舍去其它功能,打开所有的应用程序访问规则,即任何过来的数据包都可以通过 Xfilter 防火墙,同时重新扩展 Xfilter 的功能,即在 Xfilter 防火墙上实现 VPNAgent 系统中的 StaticAgent 的验证签名的功能,签名有效地数据包即可通过,签名无效的数据包不可通过。这将通过扩展 Xfilter 源代码中的 CcheckAcl 类来实现。在扩展子类中添加函数 CheckSignature,让其完成验证签名的功能。

下面主要介绍 VPNAgent 系统的数字签名和签名验证过程。

本系统采用公用密钥加密和哈希算法(即安全散列函数)来进行数字签名。公钥加密体制使用一对密钥,一个是公开的,称为公钥,另一个是保密的,称为私钥,公钥和私钥在数学上是相关的,用公钥加密的数据只能用私钥解密(用于数据加密),而用私钥签名的数据只能用公钥验证(用于数字签名)。流行的哈希算法有 MD5,SHA(可提供不同的安全级别)等。[5] 本系统选择 SHA1 算法。数字签名的过程(如图 6 上半部分)是首先对要签署的文件内容使用哈希算法,创建哈希摘要,然后签字人用私钥加密,创建他的数字签名,收件人受到经过数字签名的文件和数字签名后,对此签名进行鉴定(如图 6 的下半部分)。

首先用 JAVA 里的 KeyPairGenerator 创建公钥/私钥:

```
//用 KeyPairGenerator 创建一对公钥/私钥
KeyPairGenerator pairGen =KeyPairGenerator. getInstance(" RSA") ;
```

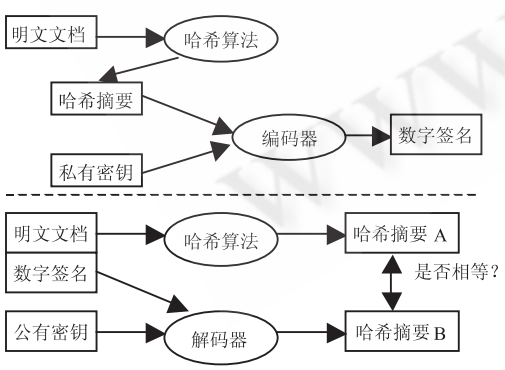

图 6　数字签名和签名验证过程

```
//用一个可信任的随机数据源创建密钥对
SecureRandom random = new SecureRandom() ;
PairGen. initialize(1024,random) ;
KeyPair pair = pairGen. genKeyPair() ;
RSAPrivateKey pvtKey = ( RSAPrivateKey) pair. getPrivate() ;
RSAPbulicKey pubKey = ( RSAPbulicKey) pair. getPublic() ;
//用序列化技术转换并保存已加密的私钥
ByteArrayOutputStream baos = new ByteArrayOutputSteam() ;
ObjectOutputSteam out = new ObjectOutputSteam(baos) ;
out. writeObject(pvtKey) ;
byte[] pvtKeyBytes = baos. toByteArray() ;
```

数字签名时,Signature 类的 sign 方法可以使我们不用操心中间步骤而直接生成签名,实现数字签名如下:

```
//已得的私钥放在 pvtKey 中,文件内容放在 strDocument 中
byte[] arDocument = getBytes() ;
//用哈希算法 SHA1withRSA 计算指定字节数组的哈希值,并对计算所得的哈希值签名
Signature signer = Signature. getInstance ( " SHA1wi thRSA") ;
signer. initSign. init(pvtKey) ;
//针对文件内容计算签名
signer. update(ardocument) ;
byte[] signatureBytes = signer. sign() ;
```

验证数字签名时,先从密钥服务器取得签字人的公钥,然后比较用公钥解密出的哈希摘要和从文件中用相同的哈希算法生成的哈希摘要。由于 Xfilter 的源代码实现用的是 VC,所以验证签名的程序用 VC 编写。

```
//连接默认的 CSP(加密服务器),接受它的句柄放入 hProv
CryptAcquireContext ( &hProv, NULL, NULL, PROV _ RSA_FULL,0) ;
//得到公共密钥的句柄放入 hPublicKey
CryptGetUserKey ( hProv, AT _ SIGNATURE, &hPublicKey) ;
//创建一个散列对象,得到它的句柄放入 hHash
```

(下转第 72 页)

(上接第 17 页)

```
CryptCreateHash(hProv,CALG_SHA1,0,0, &hHash);
    //根据文件内容计算散列值,sourceFile 是一个文件流,pBuffer 是存放读文件内容的缓冲区,BUFFER 是缓冲区的大小
    Do {
    dReadLen = sourceFile—> Read (pBuffer, BUFFER);
    CryptHashData(hHash, pBuffer,dReadLen,0);
    } while(!dReadLen < BUFFER);
    if (! CryptVerifySignature (hHash, signatureBytes, signatureBytesLen,hPublicKey,NULL,0))
    {if(GetLastError() = = NTE_BAD_SIGNATURE))
    {ShowMessage("文件已被修改") 拦截数据包 ;}
    else
    {ShowMessage("文件没有被修改");
    允许数据包通过防火墙 ;}}
```

参考文献

1 Daniel Wan. Distributed Firewall. 2001.

2 朱雁辉. Windows 防火墙与网络封包截获技术. 北京:电子工业出版社,2002:101 – 221.

3 北京启明星辰信息技术有限公司. 防火墙原理与实用技术. 北京:电子工业出版社,2002:151 – 245.

4 张云勇,刘锦德. 移动 Agent 技术. 北京:清华大学出版社,2003:51 – 201.

5 王红,曾广周,林守勋. 一种高效的移动 Agent 控制机制. 计算机工程和应用,2002,(2):250 – 252.

6 周健. 基于移动代理的网络管理框架研究. 计算机应用,2002,(5):48 – 50.

7 FARINACCI D, LI T, HANKS S. Generic Routing Encapsulation (GRE). RFC 2784 ,2000.