[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 29/06 (2006.01)

[12] 发明专利说明书

专利号 ZL 96197998.4

[45] 授权公告日 2006 年 6 月 21 日

[11] 授权公告号 CN 1260934C

[22] 申请日 1996.3.20 [21] 申请号 96197998.4
[30] 优先权
[32] 1995.10.31 [33] US [31] 08/551,260
[86] 国际申请 PCT/GB1996/000664 1996.3.20
[87] 国际公布 WO1997/016911 英 1997.5.9
[85] 进入国家阶段日期 1998.4.30
[71] 专利权人 国际商业机器公司
地址 美国纽约
[72] 发明人 罗伯特·塞西尔·戈尔
约翰·弗雷德里克·霍
审查员 陈 谦
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 王以平

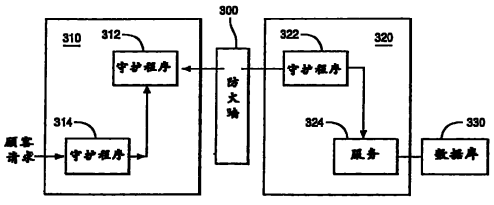
权利要求书 3 页 说明书 12 页 附图 4 页

[54] 发明名称

防止破坏安全防火墙的方法

[57] 摘要

允许外部网络(310)顾客启动一个使用内部网络(320)上内部业务资源(330)的特许业务事务而不破坏安全防火墙(300)的方法，包括认证内计算机系统(320)与外计算机系统(310)之间的连接；由外计算机系统调用所收到的事务请求(2)，建立一个包括事务请求(2)、参量和用于执行事务请求(2)的过程环境变量的串，向内计算机系统传送；由内计算机系统验证事务请求(2)，重建初始过程环境；执行事务请求(2)从而产生输出。



1. 一种用于操纵内计算机系统（320）以允许外计算机系统（310）启动一个使用内部资源（330）的事务请求（2）而不破坏内计算机系统和外计算机系统间的安全防火墙（300）的方法，其特征在于包括以下步骤:

认证（451）由内计算机系统启动的内计算机系统与外计算机系统之间的连接，从而建立一条认证的连接;

由外计算机系统调用（461）一个外计算机系统收到的事务请求;

由外计算机系统对事务请求的调用作出响应，建立（462）一个包含过程环境变量的初始过程环境，及建立一个包括事务请求和用于执行该事务请求的过程环境变量的串;

由外计算机系统通过认证的连接传送（435，436，437）串至内计算机系统;

由内计算机系统验证（454）该事务请求;

由内计算机系统重建（471）初始过程环境;

以及

由内计算机系统执行（472）该事务请求，从而生成（482）一个输出。

2. 如权利要求 1 中所要求的方法，还包括以下步骤:

(a) 由外计算机系统（310）读取（312）第一密码和第一通信端口;

(b) 由外计算机系统在第一通信端口处建立（312）第一插口并在该第一插口处侦听来自内计算机系统（320）的连接调用;

(c) 由内计算机系统读取（322）第二密码和第二通信端口；以及

(d) 由内计算机系统在第二通信端口处建立第二插口并通过第二插口发送一个连接调用至外计算机系统，从而建立（322）一个连接。

3. 如权利要求 2 中所要求的方法，其中认证步骤（451）包括以下步骤:

(e) 由内计算机系统（320）通过第二插口发送独特的时标至外系统（310）;

(f) 由外计算机系统将第一密码与收到的时标搅拌;

(g)用加密算法将搅拌的第一密码加密，从而建立第一密码串;

(h)由外计算机系统传送第一密码串至内计算机系统;

(i)由内计算机系统使用第二密码重复步骤(f)至(g)，从而建立第二密码串；以及

(j)由内计算机系统将第一密码串与第二密码串进行比较。

4. 如权利要求 3 中所要求的方法，其中搅拌步骤包括以下步骤:

将时标与十六进制数 0177 进行逻辑“与”运算以产生一个独特结果;以及

将该独特结果与第一密码的每个字符进行逻辑“异”运算从而产生搅拌的第一密码。

5. 如权利要求 3 中所要求的方法，其中加密步骤包括以下步骤:

用关键码将搅拌的第二密码中每个字符加密，从而建立密码串。

6. 如权利要求 1 中所要求的方法，其中调用步骤（461）包括以下步骤:

由外部网络发送事务请求（2）至外计算机系统（310），其中该事务请求包含输入的数据，参量，和一条用于执行事务程序的命令；以及

对外计算机系统收到的事务请求作出响应，由第一守护程序确定(312)包含过程环境变量的过程环境。

7. 如权利要求 6 中所要求的方法，还包括以下步骤:

对由外计算机系统(310)调用的事务请求(2)作出响应，调用(461)该命令;

对命令的调用作出响应，执行（462）程序原本，向它传送用户输入数据，参量和事务请求；以及

由程序原本建立（462）串，其中该串包括命令，参量和用于执行该事务请求的环境变量。

8. 如权利要求 7 中所要求的方法，还包括以下步骤:

由程序原本调用一个位于外计算机系统(310)上的客户子程序(463)，将用户输入数据，用于连接至位于外计算机系统上的第二守护程序（314）的第三通信端口，用于标识事务请求（2）类型的标识符和该串传送给它。

9. 如权利要求 8 中所要求的方法，还包括以下步骤:

由第二守护程序(314)对由程序原本收到的调用作出响应，认证(460)该客户子程序;

由第二守护程序执行分叉操作，将第三插口连接传送至子过程，以便父过程在第一插口连接处侦听来自内计算机系统（320）的调用；以及

由子过程对客户子程序的认证作出响应，传送事务请求类型至位于内计算机系统上的第三守护程序。

10. 如权利要求 9 中所要求的方法，其中验证步骤包括以下步骤:

由第三守护程序（322）读取（453）外计算机系统的存储器中存放的一个有效服务表；以及

将自子过程收到的事务请求类型与有效服务表相比较（454），其中如在有效服务表中找到该类型，则事务请求得到验证。

防止破坏安全防火墙的方法

本发明涉及供网络用的安全的网关接口，及更具体地涉及但不限于一种安全的网关接口，用于允许外部网络用户启动一个使用内部资源的特许事务而不破坏安全防火墙。

图 1 阐述一个现有技术安全的网关接口（SGI）9，它具有用于处理来自外部网络（例如 INTERNET）上任何用户的用户/顾客请求 2 的外服务器 4（例如超文本传送协议守护器 HTTPD）和用于处理来自内部网络上的任何用户（例如为具体公司工作的任何人）的请求的内服务器 7 和内数据库 8。SGI 9 还包括用于阻止从外部启动内数据库 8 上内部事务的防火墙 6。因此防火墙 6 禁止外部顾客对内部网络（即内服务器 7 和内数据库 8）启动直接连接。此限制禁止有效事务，例如产品和服务购买请求，因顾客就是不能自外部网络启动内部事务。

对上面描述的问题的常规解法包括在防火墙 6 中开一个供内向通信用的特定端口（例如端口 84）。然而此解法明显地使内部网络易受外部攻击。另一个解法是将全部所需资源（即数据库 8）放置于外服务器 4 上。然而，此解法仍然禁止执行内部事务。此外，外服务器 4 可能没有足够大的存储器以保持全部所需资源或者资源可能保密性太强而不能放置于外服务器上（如顾客数据），因而限制可提供的服务。

因此十分需要一种技术，用于允许顾客启动一个使用内部业务资源的特许业务事务而不破坏安全防火墙。

本发明的一个目的是建立一个对用户和实际事务程序都透明的安全的网关接口。

另一个目的是允许用户有效地通过防火墙向使用内部资源的内部网络启动一个事务。

又一个目的是允许用户只启动特许事务的有效集合。

还有一个目的是在外计算机系统收到来自用户的事务请求之前安全地特许内计算机系统与外计算机系统之间的一个连接。

再有一个目的是将事务程序存放于防火墙内部而不必修改它们。

按照本发明的一个方面，提供一种由计算机实施的方法，独特地编程的计算机系统和包括计算机可读程序装置的制造产品允许外部网络顾客启动一个使

用内部网络上的内部业务资源的特许业务事务而不破坏安全防火墙。

具体地说，该方法操作一个内计算机系统以允许一个外计算机系统启动一个使用内部资源的事务请求而不破坏内计算机系统与外计算机系统之间的安全防火墙。该方法包括对由内计算机启动的内计算机系统与外计算机系统之间的连接进行认证，从而建立一个认证的连接的第一步。第二步包括由外计算机系统调用一个外计算机系统收到的事务请求。第三步包括响应于对事务请求的调用，由外计算机系统建立一个包括事务请求和用于执行事务请求的过程环境变量的串。第四步包括由外计算机系统通过认证的连接将串传送至内计算机系统。第五步包括由内计算机系统确认该事务请求。第六步包括由内计算机系统重建初始过程环境。最后一步包括由内计算机系统执行事务请求从而生成一个输出。

本发明还提供一种用于操纵内计算机系统以允许外计算机系统启动一个使用内部资源的事务请求而不破坏内计算机系统与外计算机系统之间的安全防火墙的独特地编程的系统，其特征在于包括:

用于认证一个由内计算机系统启动的内计算机系统与外计算机系统间的连接从而建立一个认证的连接的装置；外计算机系统用于调用外计算机系统收到的事务请求的装置；外计算机系统用于对事务请求的调用作出响应，建立一个包含过程环境变量的初始过程环境的装置，及用于建立一个包括事务请求、参量和用于执行事务请求的过程环境变量的串的装置；外计算机系统用于通过认证的连接传送串至内计算机系统的装置；内计算机系统用于验证事务请求的装置；内计算机系统用于重建初始过程环境的装置；以及内计算机系统用于执行事务请示从而产生输出的装置。

现参照附图只通过例子描述本发明，附图中:

图 1 阐述用于实施本发明的常规网络系统的框图;

图 2 阐述用于实施本发明的代表性硬件配置;

图 3 阐述根据优选实施例的安全的网关接口（SGI）的框图；以及

图 4 阐述先前在图 3 中阐述的 SGI 的更详细过程流程图。

优选实施例包括一个由计算机实施的方法，一个独特地编程的计算机系统和一个存储器，该存储器所包括的详细逻辑用于操作一个内计算机系统以允许一个外部用户/顾客启动一个使用内部业务资源的特许业务事务而不破坏安全防火墙。

本发明实现于图 2 中阐明的计算机系统上。计算机系统 100 包括中央处理单元（CPU）10，例如用于处理高速缓存 15 的 IBM 的 Power

PC 601 或 Intel 的 486 微处理器，随机存取存储器（RAM）14，只读存储器 16 和非易失性 RAM（NVRAM）32。由 I/O 适配器 18 控制的一个或更多盘 20 提供长期存储。可采用其它不同存储介质，包括带，CD－ROM，和 WORM 驱动器。也可提供可卸存储介质以存放数据或计算机过程指令。（IBM 和 Power PC 是国际商业机器公司的注册商标和 Intel 是英特尔公司的注册商标）。

合适的台式机操作系统，如 Sun Solaris，微软的 Windows NT，IBM 的 OS/2 或 Apple 的 System 7 中来自 RAM 14 的指令和数据用于控制 CPU 10。因此台式机由 RAM 14 运行。然而在优选实施例中，IBM RISC System/6000 运行 AIX 操作系统，它是 IBM 公司实现的 UNIX 操作系统。如前所描述的，熟悉技术的人明确地知道可利用其它硬件平台和操作系统实施本发明。（Solaris 是 Sun 微系统公司的注册商标，Windows NT 是微软公司的注册商标，System 7 是苹果计算机公司的注册商标及 OS/2, RISC System/6000 和 AIX 是 IBM 公司的注册商标。UNIX 是在美国和其它国家内唯一得到 X/Open 有限公司许可证的注册商标）。

用户通过由用户接口适配器 22 控制的 I/O 设备（即用户控制设备）与计算机系统 100 通信。显示器 38 向用户显示信息，而键盘 24，指示设备 26 和扬声器 28 允许用户操纵计算机系统。通信适配器 34 控制此计算机系统与其它连至网络的处理单元间的通信。显示适配器 36 控制此计算机系统和显示器 38 之间的通信。

图 3 阐明根据优选实施例的安全的网关接口（SGI）的框图和过程流。SGI 位于一对服务器 310 和 320 上，每个服务器在计算机系统 100 上实施（见图 2）。外服务器 310 位于防火墙 300 之外，而内服务器 320 位于防火墙 300 之内。防火墙 300 利用任何合适的常规防火墙实施，该常规防火墙阻止外部事务通过它进入内服务器 320。在优选实施例中，防火墙 300 是一个网络路由选择器（如 Cisco 路由选择器）。然而熟悉技术的人明确地知道防火墙 300 可位于内服务器 320 内部。

外服务器 310 管理与外部网络，例如 INTERNET 上的用户/顾客

的通信。然而熟悉技术的人知道可用任何类型的通信协议，例如任何公共或专用网络上的SNA或X.25。内服务器320管理内部网络例如内部集合信息网络上内部资源（如数据库330）的通信。外服务器310执行一个外守护程序312，而内服务器320执行一个内守护程序322，从而允许跨过防火墙300进行通信。守护程序是一个长期运行的计算机程序，它等待外部事件并当任何时候发生这些事件时执行一系列预定动作。守护程序侦听服务请求并在被请求时完成这些请求。外服务器310也执行守护程序314，后者侦听来自外部网络的服务请求。内服务器320包括用于执行所需内部事务的服务程序324。服务程序324和内数据库330代表一套实施业务事务的计算机程序（此处将更详细地描述）。

图4阐明图3中先前阐明的SGI的更详细的过程流。外服务器310包括任何合适的常规通信协议守护程序314， cgi-bin 415, sgi_client子程序416，和外守护程序312。外守护程序312包括用于与sgi_client子程序416和内守护程序322通信的客户/服务器软件。sgi_client子程序416包括用于与外守护程序312通信的客户/服务器软件。Cgi_bin 415是一个由守护程序314执行的软件的目录。具体地说，在此例中cgi-bin 415包括的example.pl 462是一个用于与sgi_client子程序416通信的特殊程序原本（此处将更详细地描述）。在优选实施例中，守护程序314是一个常规超文本传送协议守护器（httpd）（一般也称为web服务器）。

内服务器320包括内守护程序322，服务程序324，和cgi-bin 426。服务程序324与外守护程序312和内守护程序322通信，同时执行cgi-bin子程序（例如example.pl 480）。在此例中， example.pl 480与内部集合数据库（例如图3中的集合数据库330）通信以便授权给用户/顾客并执行业务事务。

在顾客/用户能成功地在410处请求一个事务之前，内服务器320与外服务器310必须恰当地连接。为此，外操作系统执行外守护程序312，对它标识一个通信端口和位于外服务器310上文件系统（未示

出）中的密码文件的位置。接下来，对守护程序312自密码文件中读取一个八字符密码，在所标识的通信端口处建立一个插口并在该插口处侦听来自内守护程序322的连接调用。因此外守护程序312担当服务器的角度并在430处等待来自内守护程序322的连接调用，而后者担当客户的角色。此外，外守护程序312在第二端口（守护程序312的通信端口加1）上建立一个插口并在432处等待来自sgi_client子程序416的连接要求（此处将更详细地描述）。

内操作系统执行内守护程序322，对它标识一个用于将内守护程序322连至外守护程序312的通信端口，外服务器310的宿主名称，位于内服务器320中文件系统（未示出）中的密码文件位置和位于内服务器320上文件系统（未示出）中的有效服务文件位置。接下来，内守护程序322自密码文件中读取一个八字符密码，读取服务文件并在存储器中存放一个有效服务表，在标识的通信端口上建立一个插口，及最后在450处生成一个在430处侦听的跨过防火墙300对外守护程序312的标准连接调用。因为连接是由内服务器所启动的，所以防火墙300允许该连接。

在内守护程序322与外守护程序312连接以后，内守护程序322与外守护程序312必须彼此恰当地认证。为此，内守护程序322向内操作系统启动一个调用以检索当前时标，将该时标发送至外守护程序312并等待回答的认证串。外守护程序312接收时标并将其八字符密码与由内守护程序322提供的时标搅拌（改变，下面将描述）以建立认证串，用标准UNIX加密命令（或任何合适的加密算法，如DES）将此搅拌的字符串加密，然后在431处将所得认证串送至内守护程序322。下列C语言码阐述将八字符密码与时标搅拌的过程。此“create-auth”码需要三个参量：第一个是时标（即auth_time），第二个是密码（即“cred”，它是指向密码的指针），及第三个是用以存放所生成的认证串的缓存：

```
int create_auth (time_t, char * cred, char * p)
{       char    buf[9];     /*暂时缓存*/
```

```
        int       i;
        bzero     (buf, sizeof(buf)); /*清缓存*/
        strcpy    (buf, cred);        /*将口令装入缓存*/
        /*破坏缓存中的每个字符*/
        for(i=0; i<8; i++){
           buf[i]^=(auth_time & 0177); /*逻辑“与”时间标记,
                                         然后与缓存中的每个字
                                         符进行异或; 每个循环
                                         修改时间标记*/
              auth_time>>=4;   /*移位时间标记
{
for(i=0; i<8; i++)
     if  (buf [i]==0)  /*因为有效字符串不能含有空字符, 将所有
                         空字符改为 1*/
           buf[i]=1;
strcpy (p, crpyt (buf, "aa")+2); /*对于密钥利用 aa 加密缓存*/
                                 /*跳过加密结果的前两个字符(是
                                  密钥 aa)*/
                                 /*将加密结果复制到 P 所指的用
                                  户应用缓存*/
  return 0;
}
```

同样地内守护程序 322 将其密码与时标搅拌,将它加密并将它与由外守护程序 312 提供的认证串比较。如两个认证串相匹配, 则执行逆过程, 类似地由外守护程序 312 认证内守护程序 322 (也即, 自外操作系统中获取一个新时标, 将该时标传送至内守护程序 322, 内守护程序 322 将其密码与新时标搅拌, 将它加密, 及将它送回至外守护程序 312 以供验证之用)。

此认证过程使用一个外部和内部守护程序 312 和 322 双方都知道的八字符密码，一个用时标随机化的字符搅拌功能和一个加密过程。由于此搅拌功能，上述过程为每次认证和每个事务产生不同的加密认证串。这显著地减少了它遭受袭击的可能性，因为偶然获取到的认证串对随后的事务是无用的。

在内守护程序 322 和外守护程序 312 已彼此认证之后，先前担当客户角色的内守护程序 322 现担当服务器角度，在 452 处等待外守护程序 312 在 453 处提供一个服务串。外守护程序 312 在标识的第二端口上建立另一个插口并在 432 处等待(侦听)来自 sgi_client 子程序 416 的连接要求。因此，外守护程序 312 担当双重角度：对于内守护程序 322 讲是伪客户（在它们之间传送信息）而对 sgi_client 子程序 416 讲是服务器。

守护程序 314 现准备接收顾客请求 410 。例如，顾客请求可以是一个在特定期货或现钞市场上购买研究信息的事务。在 410 处，顾客决定通过在其中正运行着 http 客户应用程序用户接口的顾客系统上点击特定图标或加亮句，而执行下列事务请求：

http://external_server/cgi-bin/example.pl?stockl+stock2

该 http 客户用户接口通常向用户要求详细的事务信息（如哪一个期货或现钞市场），还有交费信息（如信用卡号）。如所请求的服务只向特许用户提供，则还要求用户输入他或她的用户识别码和密码。

传送的用户输入格式决定于用于实施事务的超文本标志语言（HTML）形式的类型。有两种常规 HTML 形式：一种“GET”形式将全部用户输入置于命令行中。因此， stock1, stock2 和任何其它用户输入都成为命令行的一部分：

.../cgi-bin/example.pl?stock1+stock2+chargecardnumber+expdate

然而由于命令行应以纯净文本形式传送过网络，所以不希望将顾客的付费卡号和失效日期传送过网络。因此使用加密的“PUT”类型的 HTML 形式以便将付费卡号和失效日期安全地送过网络。在提供所有这些信息后， http 客户应用程序通过 http 将请求送至外服务器 310 的

410 处。

在 460 处，守护程序 314 根据大家知道的安装的 HTTP 认证技术认证顾客密码（例如用标准 UNIX 加密命令将顾客密码加密并将结果与位于守护程序 314 中的 http 密码文件中的密码项目相比较）。在 461 处，如用户识别码和密码是合法的，则守护程序 314 识别“PUT”形式，自动将字符流解密及建立一个合适 UNIX 过程环境。守护程序 314 包含一个众所周知的常规 http 配置文件（未示出），用于建立一个标准过程环境，包括 PATH， USERNAME， LOGNAME 和 AUTHTYPE 变量。然后， httpsvc 470 在 471 处重建此过程环境（此处将描述）。一旦建立过程环境后，守护程序 314 执行 example.pl 462（它应位于 cgi-bin 415 中），向它传送任何所需参量（如 stock1 和 stock2）和向 example.pl 462 的标准输入流传送用户输入。

假定 example.pl 462 的确位于 cgi-bin 中，如不存在防火墙 300，则 example.pl 462 将与内数据库 330 直接通信（见图 3）并完成所需事务。然而，由于的确存在着防火墙 300 并且它阻止 example.pl 462 直接与内数据库 330 通信，所以 example.pl 不是实际的事务程序。相反地，实际事务程序是位于 cgi-bin 426 内在防火墙 300 内部的 example.pl 480。因此， cgi-bin 415 包含“特殊”程序原本（例如 example.pl 462），执行它时所用命令是用于执行位于 cgi-bin 426 中的实际事务程序时的同一命令。另一替代方案是，当外服务器 310 提供许多服务及它们中的每一个服务要求一个“特殊”程序原本以便用同一方式调用 sgi_client 子程序 416 时， example.pl 462 可能是与位于 cgi-bin 415 内的单个程序原本相连的符号链（即间接文件名称引用）。重要的是顾客可用的请求限于程序原本和分别位于 cgi-bin 415 和 cgi-bin 426 中的相应事务性程序。

程序原本 example.pl 462 将自守护程序 314 送给它的所有参量都置于过程环境（例如 SGIARG1 = stock1; SGIARG2=stock2）中，将其名称（赖以调用的名称，此例中为 example.pl）置于过程环境（例如 SGICMD = example.pl）中，执行一个 UNIX env 命令（它将过程环

境变量转储）并最后将全部过程环境变量置入头部串中。例如，头部串看起来如下：

“PATH =/bin:/Usr/bin\nAUTHTYPE=PEM\nUSERNAME
=John Doe\nSGIARG1=stock1\nSGIARG2=stock2\
=nSGICMD=example.pl”。

其次，在463处，程序原本example.pl 462调用外操作系统以检索指定的第二端口（守护程序312通信端口加1），执行sgi_client子程序416，向它传送所请求服务的类型（例如httpsvc），指定的第二端口，外服务器宿主名称，头部串，和顾客的用户识别码。example.pl 462也将任何标准输入字符流（例如用户输入的文本）作为标准输入传送至sgi_client子程序416。接着，example.pl 462在469处将自sgi_client子程序416接收的任何输出量送至守护程序314。

当sgi-client子程序在463处使用传送给它的信息运行时，sgi_client子程序416建立一个与外守护程序312的认证的连接。为此，在417处sgi_client子程序416自位于外服务器310上的私有客户密码文件（未示出）中读取一个八字符密码并在指定的第二端口处建立与外守护程序312的连接，它在432处侦听来自第二插口的连接。在433处，外守护程序312建立一个它本身的复制程序并执行它（例如一个UNIX过程分叉）。父过程将插口连接给予子过程并回至430侦听来自内守护程序322的另一个调用。

在434处，子过程认证sgi_client子程序416。为此，外守护程序312也自位于外服务器310上的私有客户密码文件（未示出）中读取一个八字符密码。外守护程序312向外操作系统启动一个调用以检索一个当前时标，在432处将时标传送至sgi_client子程序416及等待回答的认证串。sgi_client子程序416接收时标及将其八字符密码与由外守护程序312提供的时标搅拌以建立一个认证串，用标准UNIX加密命令将此搅拌的字符串加密，然后在434处将所得认证串传送至外守护程序312。类似地外守护程序312将其密码与时标搅拌，将它加密，并将它与由sgi_client子程序416提供的认证串相比较。如这两个认证

串相匹配，则 sgi_client 子程序 416 即被认证。

在 419 处，如认证成功，则 sgi-Client 子程序 416 将请求的服务的类型传送至外守护程序 312 。在此例中， sgi_client 子程序 416 总是请求一个 HTTP 服务，因为 sgi-client 子程序 416 是由 HTTP 守护程序 314 所间接调用的。先前该特殊程序原本（例如 example.pl 462 ）使用一个标明请求的服务是“ httpsvc ”的参量执行 sgi_client 子程序 416 。接下来外守护程序 312 在 435 处将“ httpsvc ”服务请求传送至内守护程序 322 。

在 452 处，内守护程序 322 等待自外守护程序 312 接收的服务请求。在 453 处，内守护程序 322 接收来自外守护程序 312 的服务请求，建立它本身的复制 映象并执行它（例如 UNIX 过程分叉）。父过程将网络插口连接给予子过程并回至 450 以启动对外守护程序 312 的另一个连接。在 454 处，子过程使用位于存储器表中的有效可执行服务（例如 httpsvc ）列表和到达这些服务的全目录路径来验证请求的服务。如请求的服务不在有效服务列表之内，则它将被否决。因此，即使一个未授权用户能够通过外守护程序 312 访问内守护程序 322 ，他/她也只能限于访问位于有效服务列表内的服务。

如该服务请求有效，则在 455 处内守护程序 322 调用一个 UNIX 执行命令执行（即用新服务程序覆盖它本身而执行）请求的服务并将网络插口连接给予 httpsvc 470 。 Httpsvc 470 将一个附加环境变量加至作为外守护程序 312 名称的过程环境上。 SGI 加上附加的环境变量以便在需要时能由 example.pl 480 决定 SGI 执行 example.pl 480 而不是 http 守护程序 314 。

附带提一下，外守护程序 312 ，内守护程序 322 ， sgi_client 子程序 416 和 httpsvc 470 中每个都具有统计和错误记录文件。每个都具有查错和跟踪参量，这些参量使不同数量的信息放于错误和统计记录文件中。此外，如跟踪参量是由 sgi_client 子程序 416 设置的，则外守护程序 312 ，内守护程序 322 和 httpsvc 470 将全部在它们相应的错误记录文件中跟踪该具体事务而不必考虑初始执行每个程序时如何将跟踪

设置的。

在436处，外守护程序312将先前建立的头部传送至服务程序324，后者在471处接收它。作为响应，服务程序324将头部（它包含初始过程环境变量）释义为variable=value串并重建在example.pl 462中定义的初始过程环境。服务程序324自头部变量SGICMD=example.pl 中确定cgi-bin 426中应调用的合适程序，建立用于与example.pl 480通信的通信信道（例如管道），和在472处调用example.pl 480。在437处，外守护程序312将标准输入字符流（例如文本）传送至服务程序324。在473处，服务程序324将文本传送至example.pl 480的标准输入端。

在这点上，由于服务程序324在471处重建初始过程环境（它是初始地在462处建立的），所以example.pl 480相信它是由http守护程序314在472处执行的，而不是SGI（虽然可选方案是它可确定SGI从由httpsvc 470加至头部的附加环境变量中调用它的）。因此SGI对顾客和http守护程序314两者都是透明的，及实际事务程序位于example.pl 480中。因此位于example.pl 480中的http守护程序314和事务程序两者都不必改变。

所有信息现已齐全以供example.pl 480在481处执行数据库330上的内部事务。一旦在481处完成事务（不论成功与否），事务的输出即退回给顾客。在482处，example.pl 480自该事务接收输出并将它输送至服务程序324的管道474。在474处，服务程序324将输出传送至外守护程序312。在438处，外守护程序312将输出传送至sgi_client子程序416。在464处，sgi_client子程序416将输出传送至特殊程序原本example.pl 462。在465处，example.pl 462将输出传送至守护程序314。在466处，守护程序314将输出传送给顾客。

因此，一个由顾客启动的事务可自守护程序314安全地传送至外守护程序312，自外守护程序312至内守护程序322供在454处验证和在481处处理，最后在466处输出退回给顾客。顾客请求和文本通过防火墙300可供内部事务处理所使用，全部在SGI完全控制之下，但

仍对顾客完全透明。因为内守护程序 322 在 451 处完成认证，严格地迫使服务在 454 处供外部网络使用，及可选地在 481 处完成用户认证，外服务器 310 的妥协带来非常小的内部安全风险而根本不会造成内部网络的妥协。

使用此具体实施例，现有 http 服务器能实施 SGI 而只需对现有 cgi-bin 命令做很少或不做修改。SGI 是完全隐藏的，能自动地支持即使复杂的 http 服务器。人们可以对现有 http 服务器做极少修改而为业务事务增加附加的安全和支持。因为外部网络可用的事务（类似于 example.pl 的程序）限于分别位于 cgi-bin 415 和 cgi-bin 426 内的程序原本和事务程序，又因为内服务器 320 正常时处于严格集合控制之下而不能轻易地被内部开发程序修改，因此 SGI 也使内部开发程序在没有集合审查和同意的情况下难于将内部事务供外部顾客使用。

在本发明的变化中，一个可替代的实施例将 sgi_client 子程序 416 和外守护程序的功能包括在守护程序 314 内。这将能提供更好性能，但将使 httpd 实施例成为独占及对它的改进难于实现。

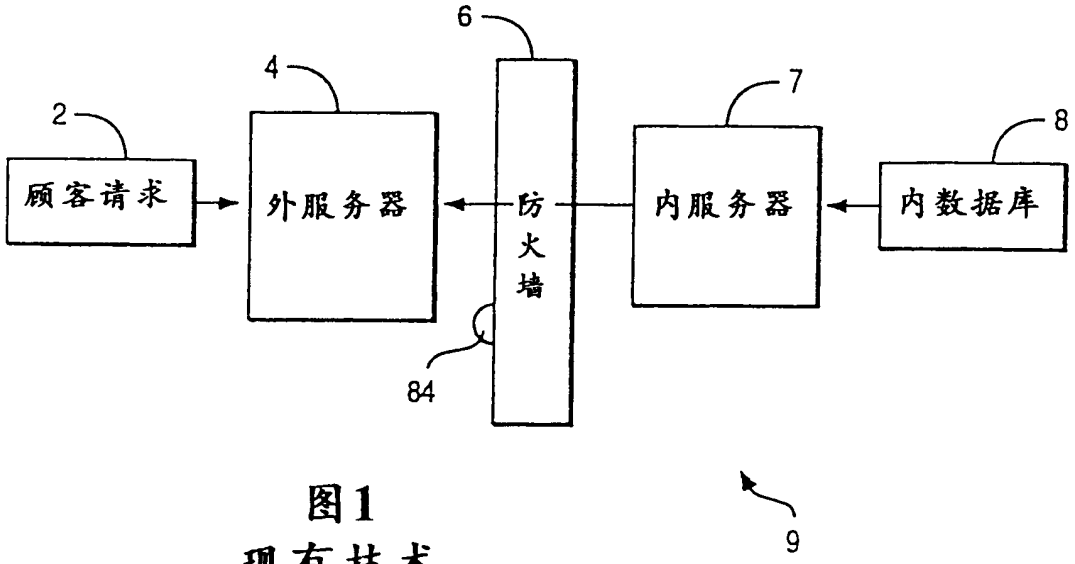

6
4
7
8
2
顾客请求
外服务器
防火墙
内服务器
内数据库
84
9

图1
现有技术

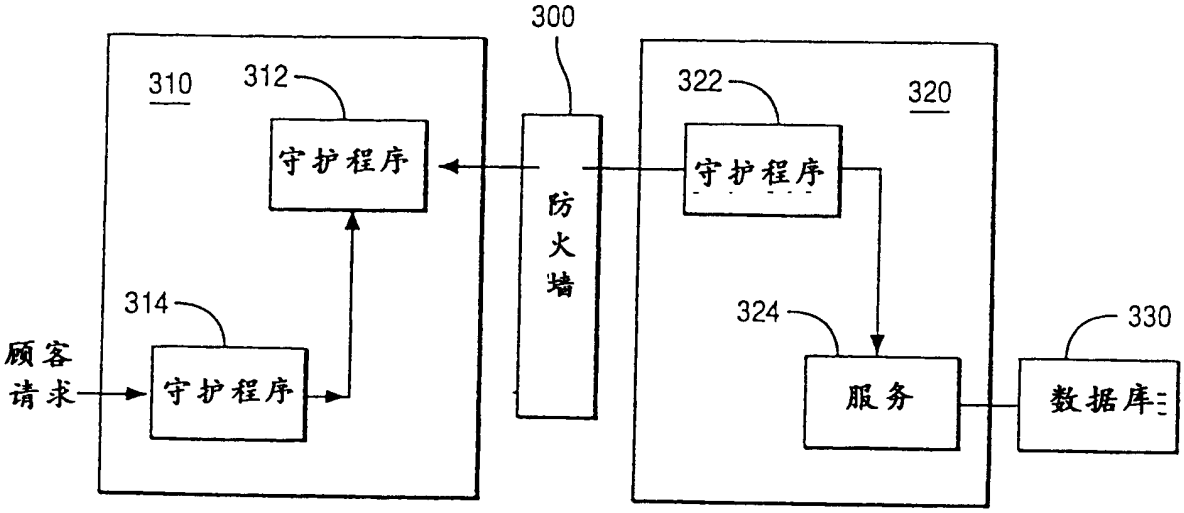

300
310
312
322
320
守护程序
守护程序
防火墙
314
324
330
顾客请求
守护程序
服务
数据库

图3

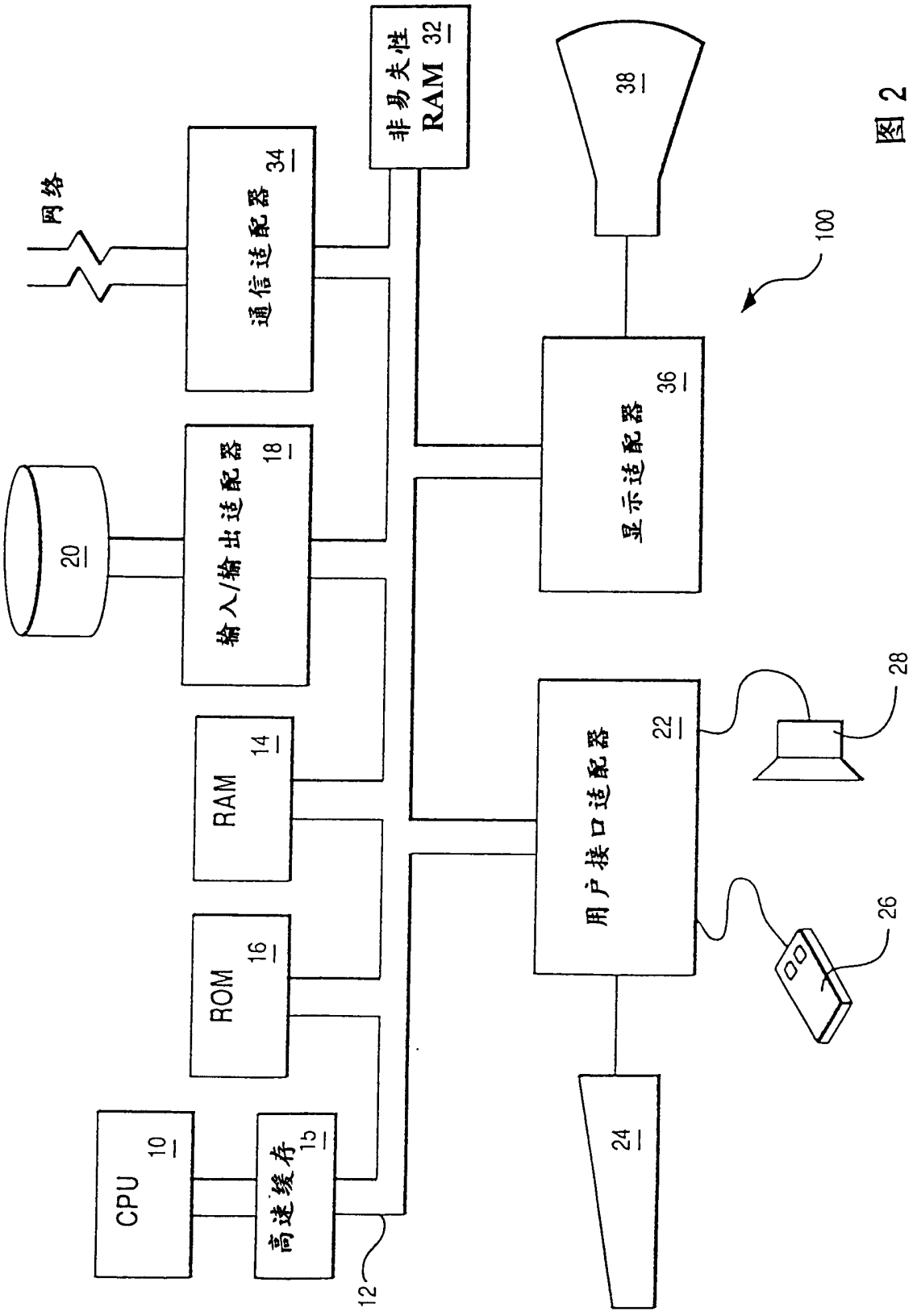

网络
通信适配器 34
非易失性 RAM 32
38
100
20
输入/输出适配器 18
显示适配器 36
RAM 14
ROM 16
用户接口适配器 22
28
26
CPU 10
高速缓存 15
12
24

图 2

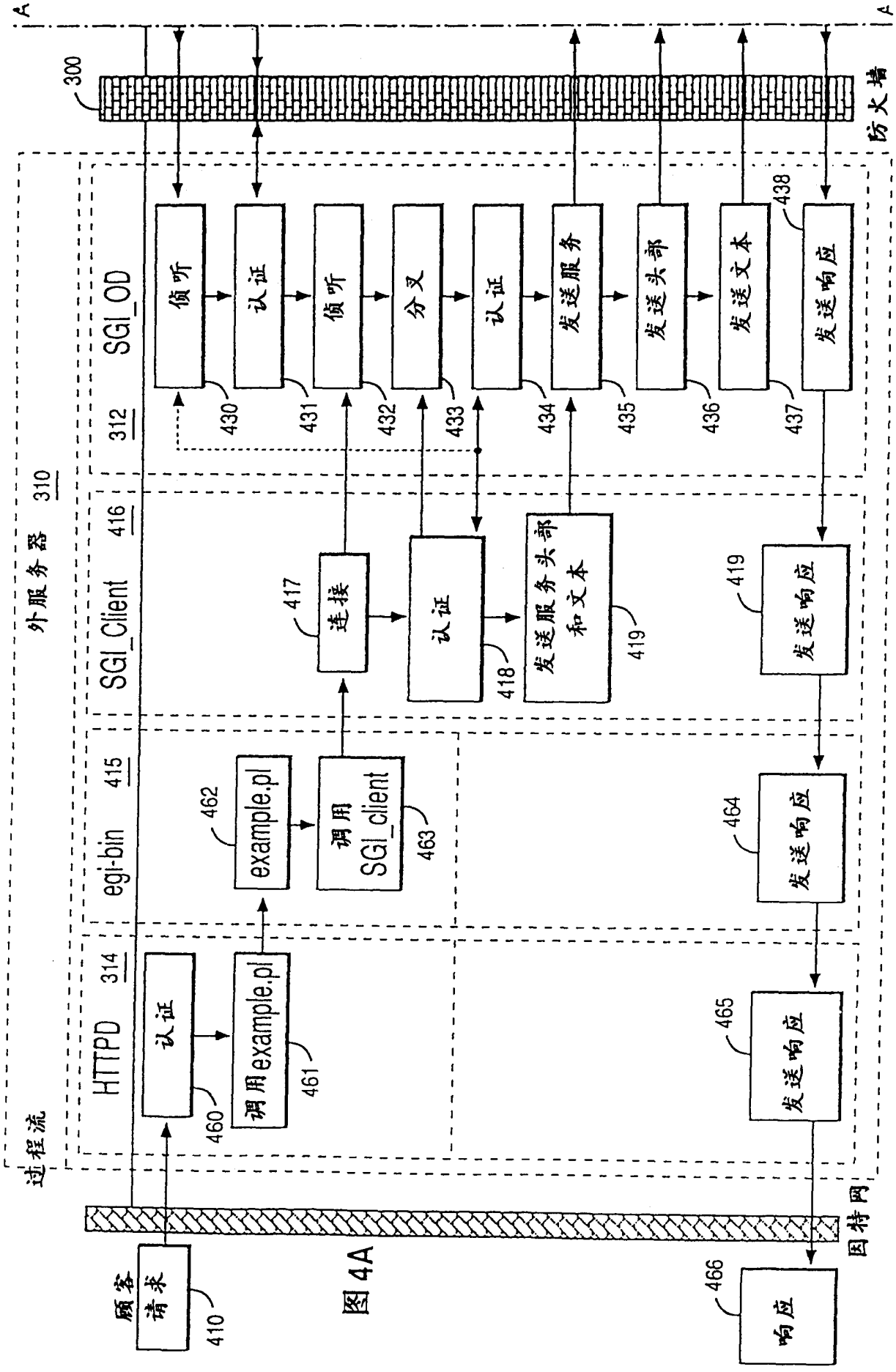
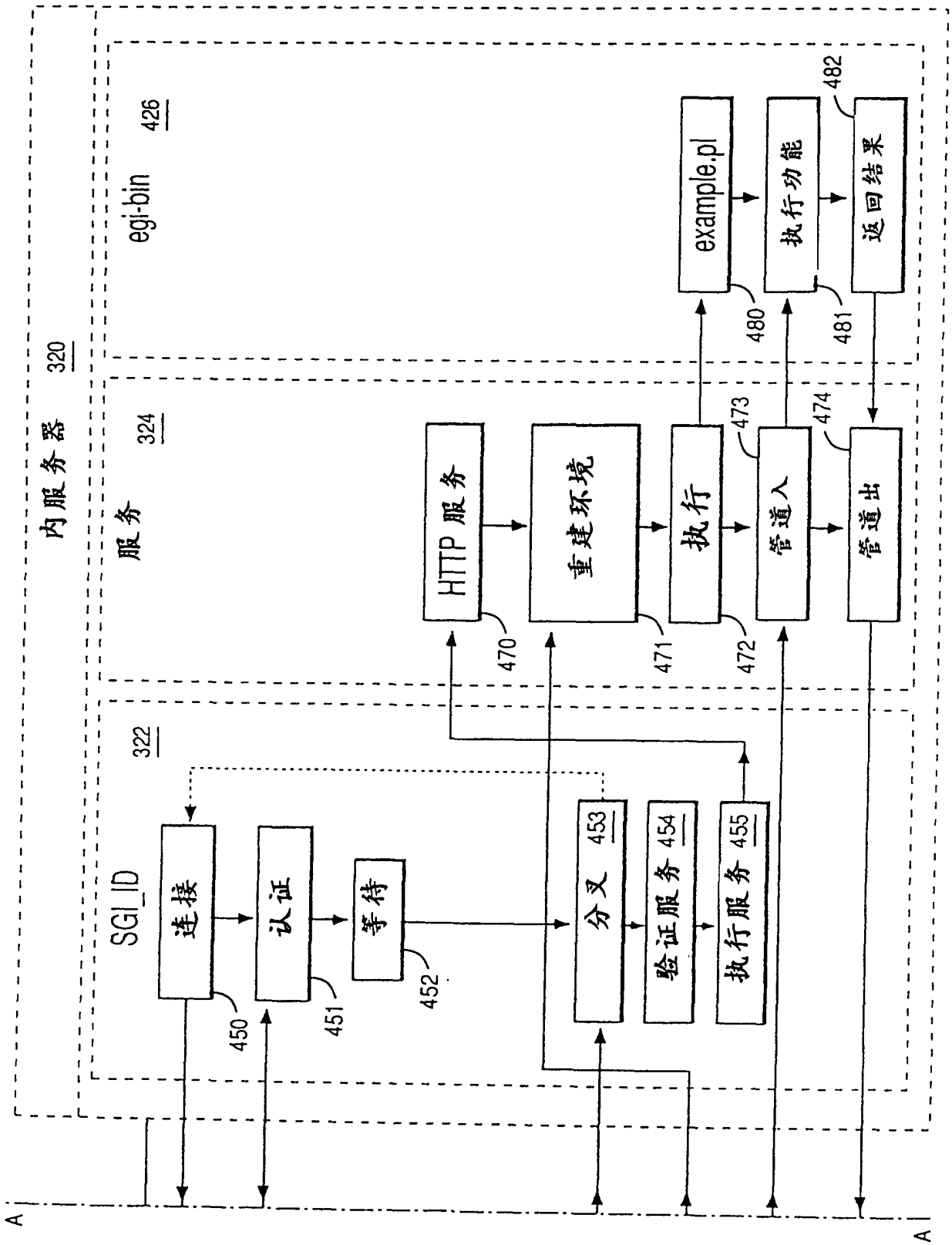

A
A
300
防火墙
外服务器 310
过程流
SGI_OD 312
侦听 430
认证 431
侦听 432
分叉 433
认证 434
发送服务 435
发送头部 436
发送文本 437
发送响应 438
SGI_Client 416
连接 417
认证 418
发送服务头部和文本 419
发送响应 419
egi-bin 415
example.pl 462
调用SGI_client 463
发送响应 464
HTTPD 314
认证 460
调用example.pl 461
发送响应 465
因特网
顾客请求 410
响应 466

图 4A

图 4B


内服务器 320
服务 324
egi-bin 426
SGI_ID 322
连接
认证
等待
分叉 453
验证服务 454
执行服务 455
HTTP 服务
重建环境
执行
管道入
管道出
example.pl
执行功能
返回结果
450
451
452
470
471
472
473
474
480
481
482
A
A