一次身份认证可访问多个应用服务器*

常晓林, 冯登国, 卿斯汉

(中国科学院 软件研究所 信息安全国家重点实验室,北京 100080);
(中国科学院 信息安全技术工程研究中心,北京 100080)
E-mail: fengdg@263.net
http://home.is.ac.cn

摘要: 一个企业网中的应用服务多种多样,各自都提供了安全措施,这必然给用户使用和管理员的权限管理带来不便.为此,设计和开发了一套一次身份认证系统.该系统可与各个应用服务很好地集成在一起,在保证安全的前提下方便了用户的使用和权限的管理.

关 键 词: 一次一密;代理;强身份认证;重发
中图法分类号: TP393 文献标识码: A

一个企业网中有各种各样的应用服务,而且这样的应用服务还在不断地增加.各个应用服务都有自己的一套安全管理机制,包括身份认证、权限管理、页面的连贯性等.一般都是根据用户的用户名和口令来进行身份认证,从而决定用户的访问权限,也就是说,用户每次访问某一应用服务时,必须先输入用户名和口令,而且从安全方面考虑,同一用户在不同应用服务中至少口令应该不同,才能降低口令泄露的危害性,提高安全保证.这种安全的考虑导致了企业网中有多少种应用服务,用户就应该有多少个口令,用户为此抱怨不止,即便用户在所有应用服务中使用同一个口令,那么用户访问多种应用服务仍要输入多少次用户名和口令,这仍然给用户带来不便,而且在客户端软件为通用的浏览器的应用服务中,用户名和口令也只是明文在网上传输,至多做一个简单的变换.用户希望从这烦琐中解脱出来,只输入一次用户名和口令,在通过强身份认证后就可访问多个应用服务.现有的认证系统,如 M.I.T.的 Kerberos security system[1]、IBM 的 KryptoKnight system[2]、SESAME(secure european system for applications in multivender environment)[3]都只解决了一对一的认证,即客户每访问一个应用服务都要参与身份认证,例如输入自己的惟一标识或插卡;即使是著名的 SSL(security sockets layer)[4],也只是保证了网上数据传输的秘密性、非否认性、完整性,使用 SSL 的各个应用服务还需用户输入其惟一标识来决定其权限.

此外,在规划企业网时,不是所有的应用服务都是现开发的,而是集成许多已有的应用服务,而这些应用服务都有自己的权限管理方式,维护各自的应用的权限表,这给企业网的管理员带来了各种不便,只要某一个用户的角色发生变化,管理员就要修改各个应用的权限表,这不但是一件很烦琐的事情,而且易出差错,即使通过驻留在各个应用中的 Agent 来更新,也容易出错.

以上两个问题成为企业网亟待解决的问题.企业网中的应用服务按客户端使用的软件可分为两类:一类是客户端为浏览器的应用服务,另一类是客户端为非浏览器的应用服务.由于第 2 类应用服务中的 C/S 之间采用的协议不是惟一的,比较复杂,我们暂且不考虑这类服务.下面,我们针对客户端为浏览器的应用服务所存在的

* 收稿日期: 2000-07-13; 修改日期: 2000-12-05
基金项目: 国家自然科学基金资助项目(60083007);国家重点基础研究发展规划 973 资助项目(G1999035802)
作者简介: 常晓林(1970－),女,福建福州人,工程师,主要研究领域为信息安全技术;冯登国(1965－),男,陕西靖边人,博士,研究员,博士生导师,主要研究领域为信息与网络安全;卿斯汉(1939－),男,湖南邵阳人,研究员,博士生导师,主要研究领域为信息安全理论和技术.

这两个问题讲述我们设计的方案.

1 身份认证解决方案

1.1 问题分析

我们知道,浏览器与服务器之间采用 HTTP 协议[5]进行通信,常用的传递数据的方式有两种:GET 和 POST.通常,WEB 数据服务器采用 POST 方式传送用户名和口令,图 1 是一个典型的 HTTP 请求身份认证包,其中最后一行的 login 和 passwd 均为变量名,Web 应用模块通过获取这两个变量后面的字符串(e.g. changxl 为用户名,gao123 为口令)来鉴别当前用户的身份.

```
Post http://login.yahoo.com/config/login?dbfsschmq8oun    HPP/1.0
Referer:
http://edit.my.yahoo.com/config/mail?&I=IWZkZHNKIXRzZnNydEpidXV4dSF0Zeq0ZHh4fH
5iu356YmNYcnM%3d&.rand=0.926280374566
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.7[en] (Win95:I)
Host: login.yoho.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-charset: iso-8859-1,*,utf-8
Cookie:B=csrjme4elft4j
Content-type: application/x-www-form-urlencoded
Content-length: 101

.tries=&.src=ym&.last=&promo=&lg=&.bypass=&partner=&.chkP=Y&.done=&login=
changxl&passwd=gao123
```

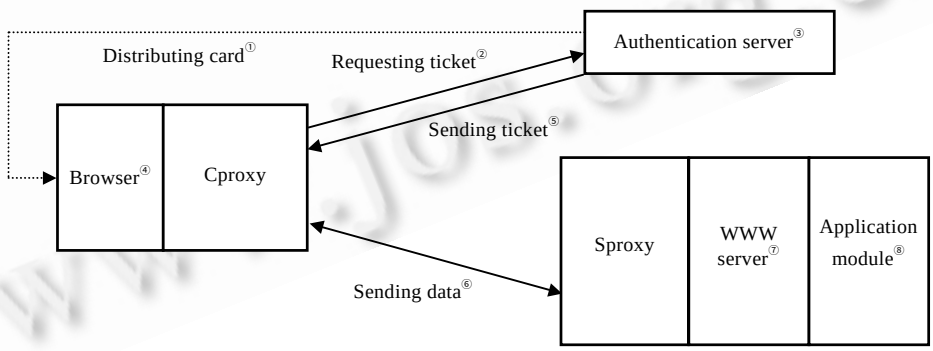
Fig.1 A typical HTTP pack for request identity authentication
图 1 一个典型的 HTTP 请求身份认证包

1.2 解决办法

我们在设计方案时暂不考虑权限管理,而是在保证安全的前提下,使得各个应用的修改降到最低限度.

1.2.1 系统模型

认证系统的物理组成部分为认证服务器、应用服务器和用户端.系统模型如图 2 所示.



①发卡,②请求票据,③认证服务器,④浏览器,⑤发送票据,⑥发送数据,⑦WWW 服务,⑧应用模块.

Fig.2 System model
图 2 系统模型

分别在用户端和应用服务器端加载一个软件代理.Cproxy(用户端代理)接收用户端浏览器的所有请求,经处理后转发给 Sproxy(应用服务器端代理),Sproxy 将处理的访问请求还原,然后将还原的请求以及用户的标识转发给 Web 数据服务器上的 WWW 服务和应用模块,由 WWW 服务返回的数据则经过 Sproxy 和 Cproxy 的处理,然后传给浏览器.

1.2.2 身份认证的流程

当 Cproxy 接到来自浏览器的任意请求时,先判断是否为 Cproxy 启动后接收的第 1 个请求,如果是,则 Cproxy 必须先去认证服务器进行身份认证,如果合法,认证服务器就发给该用户一个临时 ticket,以后用户访问某一个应用服务时,在需要输入用户名和口令的页面上直接按确定即可.当 Cproxy 接收到请求时(如图 3 所示),则对该请求作处理,包括加上 ticket,然后转发给 Sproxy.Sproxy 验证 ticket 的合法性之后,将 HTTP 请求包恢复成如图 1 所示的格式,然后转发给后台应用,这样可以保证用户的用户名和口令不在网上传输.对于其他包,Cproxy 和 Sproxy 做加密、解密、转发.现在需要解决的是,Cproxy 如何识别哪些包是身份认证请求包,哪些包只需简单地加密转发.

```
Posthttp://login.yahoo.com/config/login?dbfsschmq8oun      HPP/1.0
Referer:
http://edit.my.yahoo.com/config/mail?&I=IWZkZHNKIXRzZnNydEpidXV4dSF0Zeq0ZHh4
fH
5iu356YmNYcnM%3d&.rand=0.926280374566
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.7[en] (Win95:I)
Host: login.yoho.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-charset: iso-8859-1,*,utf-8
Cookie:B=csrjme4elft4j
Content-type: application/x-www-form-urlencoded
Content-length: 101

.tries=&.src=ym&.last+=&promo=&lg=&.bypass=&partner=&.chkP=Y&.done=&login
=passwd=
```

Fig.3 Identity authentication process
图 3 身份认证的流程

为此,我们要求应用服务必须提供其传送用户名和口令的变量名,将用户名和口令放入 Cproxy 和 Sproxy 的关键字库之中,而且将其用于身份认证的页面的文件名取为我们规定的文件名,其实客户端为浏览器的应用服务(如果仍保留其原来的管理方式)要与我们的系统集成,也只需做这点修改.因此,一个 HTTP 请求包如果以 POST 开头,页面名为规定的文件以及包含关键字,Cproxy 则可认为是身份认证包.

现在的关键是如何设计一套协议来实现认证和密钥的交换.

1.3 协议流

1.3.1 系统的初始化

(1) 为各个 Sproxy 生成公钥证书.

(2) 生成用户卡:每个合法用户的物理标识——身份卡——都由认证服务器(authenticated server,简称 AS)统一分发,发卡时 AS 保留的信息如下:

用户惟一标识 (ID)	用户名 (Username)	用户口令 (Password)	最新的使用时间 (NtimeAs)	用户公钥 (PublicKey)

其中 ID 为用户在整个系统中的惟一标识,不能修改,而其他项都可以改动.发卡时字段 NtimeAS 记录的是发卡时间,以后每次认证后记录的就是用户最新的认证时间.发卡后在卡中存放的信息如下(其实 AS 的公钥可以由 AS 传给用户,但由于在整个专用系统中,认证服务器的权力最大,因此,用户无法验证网上传来的证书就是 AS 的,所以采用将 AS 的公钥直接存入卡中):

用户惟一标识 (ID)	用户私钥 (PrivateKey)	用户公钥 (PublicKey)	最新的使用时间 (NtimeC)	AS 公钥 (ASPublicKey)

(3) 服务器的 SP 的初始化

应用服务器的 SP 的信息如下:

用户惟一标识 (ID)	用户名 (Username)	用户口令 (Password)

1.3.2 协议内容

(1) 符号说明:

AS:认证服务器;

WDS:应用服务器;

C:用户;

ID_x:X 的 IP 标识符,X 为 AS,WDS,C;

IP_x:X 的 IP 地址,X 为 AS,WDS,C;

SaltN:用户产生的随机数(C 中使用);

P_x:X 的公钥,X 为 AS,WDS,C;

P_x^{-1}:X 的私钥,X 为 AS,WDS,C;

$CERT_x$:X 的证书,X 为 AS,WDS,C;

NtimeAS:用户最新通过认证服务器认证的时间(AS 中使用);

LtimeAS:用户上次通过认证服务器认证的时间(AS 中使用);

NTimeC:用户卡中存储的上次通过认证服务器认证的时间(C 中使用);

ValidTimeLen:票据的有效时间(Ticket_data 中使用);

$K_{session}$:用对称算法加密数据的密钥;

$H(X_1,X_2,\ldots)$:表示 $X_1,X_2,\ldots$的杂凑值;

$MAC_k(X_1,X_2,\ldots)$:表示以 k 为密钥的 $X_1,X_2,\ldots$的消息鉴别码;

$(X_1,X_2,\ldots)$Key:表示用密钥 Key 加密 $X_1,X_2,\ldots$

(2) 协议流

(1) C→AS:用户向认证服务器申请票据(TICKET_req);

(2) AS→C:认证服务器颁发票据(TICKET_req);

(3) C→WDS:用户请求申请 Web 数据服务器的证书(CERTWDS_req);

(4) WDS→C:Web 数据服务器响应用户的证书请求(CERTWDS_req);

(5) C→WDS:用户对 Web 数据服务器的 POST 握手请求(SHAKE_req);

(6) WDS→C:Web 数据服务器响应用户的 POST 握手请求(SHAKE_req).

下面分两个阶段详细介绍协议的实现过程.

阶段 1. 用户从认证服务器获取票据

(1) 用户向认证服务器申请票据.

$$TICKET\_req:\{ID_C,SaltN\}P_{AS},Sign\_data,$$

其中 $Sign\_data=\{H\{ID_C,IP_C,NtimeC,SaltN\}\}P^{-1}{}_C$.

(2) 认证服务器获取用户的 IP_C(利用了 Socket 的特点),然后用 $P^{-1}{}_{AS}$ 解密获得 ID_C 和 SaltN,验证 Sign_data,如果验证签名不正确,则认为是非法用户.在确认为合法用户后,让 LTimeAS 等于 NTimeAS,并用系统当前时间更新表 1 中 NtimeAS 的值,然后发出响应信息:

$$TICKET\_rep:\{NtimeAS,ValidTimeLen\}K_{rand},Ticket\_data,MAC,$$

其中 $K_{rand}=MAC_{SaltN}\{ID_C, IP_C, LtimeAS, SaltN\}$, $Ticket\_data=\{H\{ID_C, IP_C, NtimeAS, ValidTimeLen\}\}P^{-1}{}_{AS}$, $MAC=MAC_{K_{rand}}\{Ticket\_data,NtimeAS,ValidTimeLen\}$.

(3) 用户接到正确响应信息后,生成 $K_{rand}=MAC_{SaltN}\{ID_C,IP_C,NtimeC,SaltN\}$,解密并验证认证服务器发来的信息,用 NtimeAS 更新卡中的 NtimeC,保留票据 Ticket_data 和 ValidTimeLen.

阶段 2. 用户与应用服务器握手.用户先去申请应用服务器的证书,并验证合法性.

(1) CERTWDS_req

(2) CERTWDS_rep:

$$RandomSP,CERT_{WDS},$$

其中 $CERT_{WDS}=ID_{WDS},P_{WDS},IssueTime,ValidTimeLen,\{H\{ID_{WDS},P_{WDS},IssueTime,ValidTimeLen\}\}P^{-1}{}_{AS}$.

(3) SHAKE_req:

$$\{ID_C,NtimeC,ValidTimeLen,RandomSP\}P_{WDS},Ticket\_data,\text{HTTP 包}.$$

(4) SHAKE_rep:

Sproxy 收到该消息后,首先解密得到 $ID_C,NtimeC,ValidTimeLen,RandomSP$,同时获得发出握手请求的用户的 IP_C,然后验证 $RandomSP$、时间和 $Ticket\_data$ 的正确性:如果上述验证和比较有一个不对,则认为是非法用户.在确认为合法用户后,Sproxy 要做:① 首先检查自己的密钥库中是否有以该 IP_C 和 $NtimeC$ 为关键字的记录,若没有则生成密钥 $K_{WDS}=H\{ID_C,IP_C,NtimeC\}$,然后删除所有含该 IP_C 的记录,然后将 $IP_C,NtimeC$ 和 K_{WDS} 写入库中,以后 Sproxy 和 Cproxy 之间就用该密钥加密数据;② 根据 ID_C 从应用服务器的 SP 的信息表中取出用户名和口令,贴上,将请求还原成如图 1 所示的形式,然后转发给 WWW 服务.

2 权限管理解决方案

因为一个企业中的角色种类的划分一般是不变的,即使发生变化也很少.我们采取的权限管理策略是:用户→角色,角色→权限.也就是说,由认证服务器按照企业结构和工作种类进行角色分类,在认证服务器建立下表:

角色标号 (Role_ID)	角色名称 (RoleName)

将该表传给各个应用服务器,然后在各个应用服务器上按角色授权,这些工作在一个企业网中发生的频率很少.同时认证服务器在发卡时 AS 保留的信息表中增加一个字段 Roles,字段 Roles 有 64 个字符(根据企业具体情况而定),共 512 位,每一位表示一个角色,可表示 512 种角色,某一位(bit)为 1,则用户表示拥有该角色,否则置为 0.认证服务器可随时更改用户拥有的角色,用户拥有的角色的集合体现在上述表中的 Roles 字段,用户每次通过认证服务器的认证后,认证服务器将用户的角色传给用户,然后用户在与各个应用服务器握手时,将自己的角色也交给应用服务器,应用服务器从而算出用户的权限.

协议流更改如下:

(1) TICKET_rep:

$$\{NtimeAS,ValidTimeLen,Roles\}K_{rand},Ticket\_data,MAC.$$

其中 $K_{rand}=MAC_{SaltN}\{ID_C,IP_C,LtimeAS,SaltN\},Ticket\_data=\{H\{ID_C,IP_C,NtimeAS,ValidTimeLen,Roles\}\}P^{-1}{}_{AS},MAC=MAC_{K_{rand}}\{Ticket\_data,NtimeAS,ValidTimeLen,Roles\}$.

(2) HAKE_req:

$$\{ID_C,NtimeC,ValidTimeLen,RandomSP,Roles\}P_{WDS},Ticket\_data,HTTP\ 包.$$

这种管理方式减轻了系统管理员的负荷,同时杜绝了由于失误带来的损失.缺点是应用服务器端的模块要做部分修改.

3 结束语

虽然上述协议是针对浏览器方式的应用服务提出的,但其不失普遍性.非浏览器的应用服务可分为两重模式和三重模式,在这两种结构中,应用服务本身提供了加密 C/S 之间数据的功能,所以只需解决一次身份认证和统一权限管理的问题.

References:

[1] Kohl, J., Neuman, C. The Kerberos network authentication service (V5). RFC1510, 1993. http://www.ietf.org/rfc/rfc1510.txt.

[2] Molva, R., Tsudik, G., van Herreweghen, E., *et al*. KryptoKnight authentication and key distribution system. In: Deswarte, Y., Eizenberg, G., Quisquater J.-J., eds. Proceedings of the 2nd European Symposium on Research in Computer Security (ESORICS'92). Lecture Notes in Computer Science 648, Springer-Verlag, 1992. 155~174. http://www.laas.fr/~esorics/notices/MTHZ92.html.

[3] Kaijser, P., Parker, T., Pinkas, D. SESAME: the solution to security for open distributed systems. Computer Communications, 1994,17(7):501~518.

[4] Fielding, R., Gettys, J., Mogul, J., *et al*. Hypertext transfer protocol—HTTP/1.1. RFC2068, 1997. http://www.w3.org/protocols/rfc2616/rfc2616.html.

[5] Freier, A.O., Karlton, P., Kocher, P.C. The SSl protocol, version 3.0, Internet-draft. draft-freier-ssl-version3-02.txt, 1996. http://wp.netscape.com/eng/ssl3/ssl-toc.html.

Access Multiple Application Servers by Using Once Identity Authentication*

CHANG Xiao-lin, FENG Deng-guo, QING Si-han

(*State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*);

(*Engineering Research Center for Information Security Technology, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: fengdg@263.net

http://home.is.ac.cn

Abstract: Various kinds of application service are used in corporation network; which provide security measures. It isn't convenient for users to use and for managers to manage limits of authority. Therefore, a set of once identity authentication system is designed and developed. The system can be well integrated with every application service, and it is convenient for users and managers under ensuring security.

Key words: once pad; proxy; strong identity authentication; re-transaction

* Received July 13, 2000; accepted December 5, 2000

Supported by the National Natural Science Foundation of China under Grant No.60083007; the National Grand Fundamental Research 973 Program of China under Grant No.G1999035802