支持跨平台应用的单点登录系统①

许 逸[1], 沈映珊[2]

[1](汕头职业技术学院 计算机系, 汕头 515071)
[2](华南师范大学 计算机学院, 广州 510631)

摘 要: 典型单点登录技术基于 Kerberos 协议, 票据存活期设置存在安全漏洞, 同时多数单点登录技术仅适用于 WEB 环境. 针对于此, 提出一种新的认证模式, 实现跨平台的单点登录系统. 该系统以 PKI 为安全基础, 使用定期报到方式解决票据和认证信息的时效性问题, 在客户端采用 Hook 和 BHO 技术, 由代理模块对登录认证进行代理. 结果表明, 该系统改进了票据存活期机制的不足, 部署时不需应用程序开放开发接口, 具有易部署、易扩展等特性.

关键词: 单点登录; 跨平台; 定期报到; 票据安全; 票据时效

Cross Platform Single Sign-on System

XU Yi[1], SHEN Ying-Shan[2]

[1](Department of Computer Science, Shantou Polytechnic, Shantou 515041, China)
[2](School of Computer, South China Normal University, Guanzhou 510631, China)

Abstract: The typical single sign-on technologies are based on Kerberos protocol. There are vulnerabilities in the ticket's lift-time solution. And most of the single sign-on technologies are applied to Web-based application systems. According to these, a new mode of authentication is proposed. And a single sign-on system is designed and implemented. The system uses PKI as security infrastructure, uses a technique called regular check-in to solve the limitation of tickets and authentication information. The client agents of the sign-on and identity authentication use Hook API and BHO. The results show that the system can overcome the shortcoming of the lifetime, without requiring the existing infrastructure support development interface, and with advantages such as flexible deployment and expansion.

Key words: single sign-on; cross platform; regular check-in; ticket's security; ticket's prescription

单点登录(Single Sign-On, 记为 SSO)是联合身份管理的重要概念[1], 旨在支持用户在一次认证之后可访问应用域中的所有网络资源和服务[2]. SSO 用于解决分散的认证授权机制中的安全和管理问题, 实质是在多个应用系统之间传递或共享信任票据[3], 要解决网络资源对首次认证凭据(即票据)的信任问题, 以及信任传递的安全问题. 使用 SSO 的应用系统具有效率高、安全性强等优良属性, 因此 SSO 系统在大型网络应用系统中得到了广泛的应用[4]. 尤其近年云计算的发展, 要求更强的身份认证机制, 而单点登录在分布式环境中具有可行的解决[5], 因此分析 SSO 的安全性, 并设计相关解决方案是网络安全领域的研究一个重点.

但是, 目前主流 SSO 技术如 CAS、JOSSO 等, 多为基于 Web 的 SSO 系统, 通过 cookie 或 session 机制来实现认证信息的共享和传递. 而多数企业环境混合了 B/S 和 C/S 体系结构的应用, C/S 应用层协议由开发商定义使用, 认证流程不存在开放的标准, 实现 SSO 时要对来自不同厂商的应用系统的认证流程, 甚至数据结构进行改造, 对已部署的应用系统不具可操作性.

同时, 典型的 SSO 技术基于 Kerberos 协议[2], Kerberos 协议通过给票据设置时间戳防止重放攻击,

① 收稿时间:2013-12-11;收到修改稿时间:2014-01-14

但是在票据存活期中仍然存在被重用的可能[6]. 票据存活期直接影响到安全性, 过长的存活期会增加安全风险. 用户登录后未注销并销毁票据, 或者意外断电导致票据残留在本地电脑, 导致其他人可能通过使用该电脑直接访问授权访问的应用; 另外, 还要防止攻击者入侵电脑直接获取票据, 在票据有效期, 通过其他机器使用. 而过短的票据存活期则导致用户增加登录频率, 无法达到单点登录的一次登录在较长时间内有效的效果. 而且, 时间戳要求整个网络(包括服务器和客户机)时钟同步, 而时钟同步机制容易受到破坏或错误, 文献[7]称之为压制重放攻击(Suppress Replay Attack).

针对上述主流技术存在的安全缺陷以及在应用中存在的不足, 本文研究一种新的跨平台的统一认证模式. 提出用定期报到(check in)技术来取代票据存活期设置, 即客户端定期向认证服务器报告其在线状态(有效认证); 同时, 在应用系统服务器端部署拦截器, 拦截器与认证服务器同步用户有效认证信息, 并根据一定策略放行客户端持票据的访问请求. 在此认证模式基础上, 本文描述一种 SSO 系统的设计、实现, 在客户端由代理模块对登录认证进行代理. 与现有工作对比, 本文所实现系统中, 票据有效性不依赖票据的时效属性, 而依赖于用户在线状态是否有效; 而用户在线状态通过向认证服务器定期报告(下称 checkin), 并采取措施防止 checkin 的重放. 最坏情况下, 票据被攻击者盗用并获取了应用服务的访问权限, 攻击者必须伪造和计算 checkin 中携带的各项参数, 使认证服务器的认证信息有效, 否则无法访问应用服务. 由于 checkin 周期进行, 其周期时间大大少于票据整个生命期所需要的时间, 因此具有更高安全性.

1　系统设计

1.1　系统模型

新设计的单点登录系统采用公钥基础设施(PKI, Public Key Infrastructure). 非对称加密技术占用计算资源较多, 仅用于认证用户的内网准入、密钥传递、以及认证服务器和拦截器之间同步数据时的数字签名; 其他关键信息的加密均采用对称加密实现. 该系统模型如图 1:

单点登录系统的认证过程: (1)用户首次登录, 在统一身份认证服务器(下文简称 AS)认证成功, AS 将用户的关联认证信息保存到 SSO 客户端; (2)SSO 客户端定期向 AS 发送在线状态信息(下文简称 checkin), AS 根据其 checkin 状态确定用户认证信息的有效期, 以此取代 lifetime+TimeStamp 的解决方式; (3)同时, 在应用系统(下文简称 APP)端布局拦截器(下文简称 Filter), Filter 向 AS 同步用户的认证状态; (4)用户访问 APP 时, SSO 客户端携 APP 的认证信息向 APP 发送请求; (5)Filter 拦截、检查并放行合法请求.

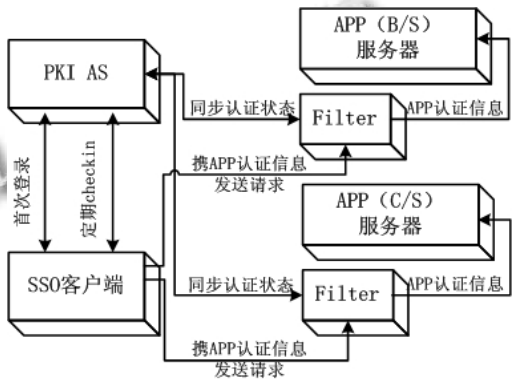

图 1　单点登录系统模型

若用户绕过 SSO 客户端访问 APP, 则会因为没有携 APP 认证信息, 或者 Filter 发现用户未认证, 而导致无法登入 APP. 由于 Filter 与 AS 同步数据, 用户无法在非认证状态下获得 APP 的访问权限. 即使个别 APP 的用户认证数据泄露, 攻击者直接使用应用客户端登录, 也会被 Filter 拦截.

其中, 对保密性和时效性的解决思路是: 使用 PKI 实现信源认证和会话密钥保密性; 使用对称加密实现票据保密以及保证会话双方身份; 使用随机数和序列号防止信息被篡改、重放; 使用定期 checkin 的方式保证票据和认证信息有效.

对跨平台问题的解决思路是: 使用 Hook 和 BHO 技术, SSO 客户端拦截并代理应用服务的认证模块, 同时在 APP 服务器端布局 Filter, Filter 根据 AS 的认证信息来拦截或允许来自用户端的请求.

1.2　系统设计细节

国内利用 PKI 机制控制用户准入内网的企业中, 大部分采用基于 USBKey 的 X.509 身份认证. X.509 身份认证的安全性取决于私钥的安全性, USBkey 内置了 CPU, 内嵌 SmartCOS-PK 操作系统, 和读写控制器, 私钥不能读出, 私钥加密在设备内完成[8]. 本系统采

用 PKI 提供认证服务, 适合直接应用于此环境. 但也可以将证书存储于本地, 此时数字证书应采用口令保护, 其安全性依赖操作系统证书管理服务的安全性.

对涉及技术缩写进行统一注解:

C: Client, 用户的 SSO 客户端.

AS: 认证服务器.

Pr: 私钥, 其中 PrC 指 SSO 客户端 C 的私钥, PrAS 指 AS 的私钥.

Pu: 公钥.

K: 对称密钥, 其中 KS 指会话密钥.

ID: 用户的 ID, IDC 指用户在 SSO 的统一 ID, IDAPP 指用户在某一应用上的用户 ID.

AD: 用户的 IP 地址.

N: 随机数.

Ticket: 票据, TicketC 指 AS 颁发给 C 的 Ticket.

IP 动态白名单: 存放在各应用服务器的一张加密表, 为有权向访问本应用的用户认证 IP 地址.

++: 参考 C++中运算符++的定义.

1.2.1 认证(首次登录)

当用户第一次登录系统时, 用户插入 USBkey, PIN 码验证通过后, SSO 客户端 C 向 AS 发起认证请求(1), AS 确认请求的真实性并向 C 传送 TicketC 和 AuthenticatorC(2):

(1)C→AS:IDC||E(PrC, [IDC||N0])

加密数据部分的 IDC, 提供了解密后识别校验 PrC 的功能.

N0 为随机数, 用于防止重放攻击. 如果 N0 明文传送, 即采用 IDC|| N0||E(PrC, IDC), 则导致如下结果: 攻击者可通过比较两次截获的数据包, 推测出 E(PrC, IDC)的长度, 并构造认证请求 IDC|| NX||E(PrC, IDC)冒充用户向 AS 发起认证请求.

(2)AS→C:TicketC||AuthenticatorC ||E(PrAS, ++N0)

TicketC= E(PuC, [KS||IDC])

AuthenticatorC=E(KS, {[IDAPP || UserID@APP || Password@APP || Status@APPFilter]})

AuthenticatorC 的{ } 中的内容重复次数为用户有访权访问的应用系统数, IDAPP 是应用系统 ID、UserID@APP 和 Password@APP 是用户在应用系统上的 ID 及口令、Status@APPFilter 是应用系统端 Filter 的工作状态: Status@APPFilter＝0/1, 0 表示 SSO 部署在 APP 前端的 Filter 工作正常, 1 表示异常, 当 AS 无法向 Filter 同步数据时, 该值置 1, 此时在客户端禁止用户对该 APP 的访问. 如下表, 某用户对 A 应用、C 应用、D 应用、F 应用有访问权限, 且由于网络故障, C 应用的 Filter 工作异常:

A 应用 ID	UserID@A 应用	Password@A 应用	0
C 应用 ID	UserID@C 应用	Password@C 应用	1
D 应用 ID	UserID@D 应用	Password@D 应用	0
F 应用 ID	UserID@F 应用	Password@F 应用	0

C 通过解密 E(PrAS, ++ N0)获得++N0 的值, 验证 AS 的真实性, 避免接收到伪造的 Ticket 和 Authenticator. 由于 C 已知 N0, 使用++N0 则无需设计特殊结构以判断随机数是否正确解密; 通过解密 E(PuC, KS)获得本次认证的会话密钥 KS; TicketC 和 AuthenticatorC 缓存在本地.

(3)C→AS:ACK=E(KS, N0++)

N0 为(2)中自增 1 后的 N0 值.

AS 收到(3), 确认用户获得 TicketC 和 AuthenticatorC, 设置用户 IDC 为 sign-on 状态, 即将 IDC 和对应的 ADC 值记入 sign-on 表, ADC 为本次认证的 IP 地址, 完成认证:

UserID	ADC	Checkin-Time
012345	192.168.2.27	2013-06-18 13:17
023456	192.168.2.79	2013-06-18 13:21
145677	192.168.2.23	2013-06-18 13:21
.....		

之后, AS 根据 UserID 的权限和 sign-on 状态, 向各应用 Filter 分别同步 IP 动态白名单(有权限访问该应用的认证用户的 IP 地址), 数据以加密方式传送:

(4)AS→APP: IDAPP|| IP 动态白名单||E(PrAS, HASH(IP 动态白名单)|ST1)

AS 和 SSO 体系中应用服务器同步时间较易实现, 因此用时间戳 ST 来防重放, 相对于随机数防重放, 系统消耗较少资源.

AS 用私钥对 IP 动态白名单的哈希值加密实现数字签名.

这里讨论一下 Filter 端是否必要向 AS 发回应确认接收(4): 如果采取回应确认策略, 当(4)丢包, 或者 Filter 发回的回应丢包/超时, AS 要重发(4), 如果网络环境不稳定或 SSO 中支持的 APP 数较多, AS 负担较大, 尤其在认证高峰时间(如果是在企业, 上班时间后一段时间内为高峰期), 容易超时而造成对 AS 的 DOS 攻击.

考虑到应用环境, 这里仅在认证成功时由 AS 向 Filter 同步数据, 不需 Filter 回应, 同时采取由应用服务器的 Filter 定期向 AS 发出 get 指令请求同步数据.

IP 动态白名单还可以实现另一种效果: 即使用户在应用系统的认证信息被攻击者获取, 但攻击者无法在 AS 上认证自己, 则访问应用系统会被 Filter 拦截.

1.2.2 Check in(定期报到)

客户端 C 在(3)完成认证后, 每隔 T 时间向 AS checkin:

(5)C→AS:checkin=IDC||E(KS, N1++)

N1, 防重放攻击, N1 的初值是(3)中 N0 自增 1 后的值, 此后 N1 由上一次的值自增 1 得来, 容易校验, 并防止攻击者在用户非正常离线时冒充用户身份 checkin 维持认证有效.

非正常离线, 指非正常断电/网络故障/非正常退出客户端. 正常离线, 指客户端注销/用户拔出 USBkey.

AS 在内收到客户端的 checkin 信息(5)后作如下处理, 若 IDC 的状态为 sign-out, 即不在 sign-on 表(上一次 checkin 不成功或用户注销), 此可能为重放包或超时包, checkin 无效; 若 IDC 已 sign-on, 更新 sign-on 表中的 checkin-time, 并解密得 N1.

AS 定时检查 sign-on 表, 若用户在 Timeout 时间内未 checkin, 则从 sign-on 表删除相应用户记录, 同时执行(4), 更新 Filter 端的 IP 动态白名单.

于此 2 同时, C 发出 checkin(5)后等待 AS 的回复:

(6)AS→C: ACK= E(KS, N1+1)

加密算法的雪崩效应使攻击者无法根据(5)中 N1 值推测出 N1+1 的加密规律.

C 在 Timeout 时间内收到 ACK, 则认证有效, 并重新对 T 和 Timeout 计时; 若 T+Δt 未收到 ACK, 则重发 checkin, 即重复(5), N1 用原值(AS 在 timeout 内收到 N1 值相同的 checkin 会忽略, checkin-time 不刷新, 避免攻击者在用户非正常离线后重发 checkin 以延长认证有效). n 次重发 checkin 无回应, 客户端主动注销, 销毁 KS、TicketC、AuthenticatorC, 提示因服务器或线路故障用户已注销认证.

以下对几个时间值及关系进行注释:

T SSO 客户端 C 在(3)认证成功或(6)checkin 成功后计时, 每隔时间 T 向 AS checkin;

Δt 客户端对一次 checkin 握手(5)和(6)的时延期望值;

Timeout AS 对客户端 checkin 间隔的最大容忍时间;

它们满足关系: T+Δt*n≥Timeout n 为设定的重发次数.

1.2.3 注销认证

(7)C→AS:signout=IDC||'FIN'

'FIN'为结束认证的标志

C 发出结束认证的请求后, 销毁 KS、TicketC、AuthenticatorC. AS 接收到请求后, 从 sign-on 表删除用户记录, 执行(4), 更新 Filter 端的 IP 动态白名单.

假设(7)丢包/被拦截, 由于客户端 C 已经注销, 不会再进行 checkin, 至多在 Timeout 时间内 AS 会获知 C 已注销; 假设 AS 对 Filter 的通知丢包/被拦截, 则 Filter 会在同步数据时再更新白名单, 具体措施见 1.2.4.

1.2.4 Filter 定期 T0 用 get 指令同步 IP 动态白名单

(8)Filter→AS:get IDAPP

(9)AS→Filter: IDAPP|| IP 动态白名单||E(PrAS, HASH(IP 动态白名单)|ST2)

同(4), AS 用私钥对 IP 动态白名单的哈希值加密实现数字签名, 提高了加密速度. 另外, 由于 get 指令同步数据是定期执行, 用户认证信息未必有改变, (9)也可处理为 IP 动态白名单暂不发送, Filter 对比哈希值, 若名单有变化, 再向 AS 请求名单, 以增加握手次数来节省网络流量.

如果(8)或(9)丢包, 导致 Filter 没有收到回应, 则 Filter 每隔时间 Δt0 重发(8), m 次后没有收到 AS 的回应(网络故障/AS 拒绝服务/与 AS 之间的通信受攻击), 清空 IP 动态白名单, 按一定的策略人工恢复或自动再尝试连接 AS.

上述 1.2 系统设计细节如图 2:

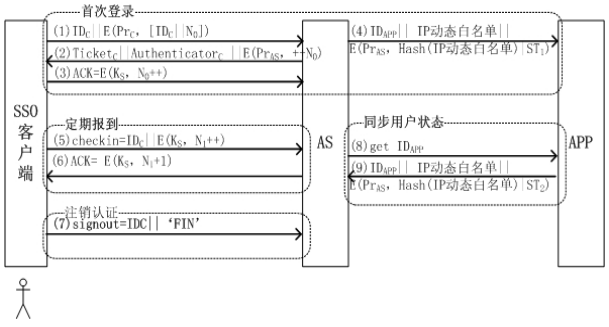

图 2　系统设计概述

1.3 用户访问 APP

当用户访问 APP 时, 客户端 C 检测到登录窗口激

活事件, 从 AuthenticatorC 中获取 APP 的认证信息并填入组件, 启动向 APP 服务器的请求; APP 服务器端的 Filter 拦截下访问请求, 获得 IP 地址, 向本地的 IP 动态白名单查询该IP地址是否允许通过, 若有则放行. 拦截器在本地查询认证数据, 拦截和查询效率高.

1.4 特殊情况处理与反馈

存在一个 IP 可能会有多个认证用户使用的情况, 尽管这样的情况很罕见, 比如某个使用者拥有两份 key. 但客户端可以根据当前用户来选择 ticketC, 并从相应的 AuthenticatorC 中获取当前用户的 APP 认证信息和权限. 如果选错了 ticketC, 则客户端无法读取用户在 APP 的认证信息, 要求用户重选.

如果一个认证用户在同一个 APP 中有两个身份, 可在客户端用ticketC解密后检测到多个认证身份而要求用户选择.

2 跨平台的解决

文献[9]提出的解决方案中, 应用的认证信息以 XML 配置文件存放在 USBkey, 用户通过客户端调用应用, C/S 应用由客户端向应用的登录窗口填入认证信息并激活提交, B/S 应用则使用配置文件中的数据构造 GET 或 POST 方式的提交数据实现登录, 后者需针对客户端加密认证信息的应用单独分析加密方法并个别处理, 存在较大难度. 而且配置文件存放于 USBkey, 且能被读出, 存在遗失和盗用风险. 为实现对用户透明, 在解决跨软件结构时, 考虑采用 Hook 和 BHO 解决.

Hook 是 Windows 提供的一种替换 DOS 下中断的系统机制. 在对特定事件进行 Hook 后, 该事件发生时, Hook 机制可使截获 Windows 系统内部传递的消息, 并在目标窗口之前处理它[10], 包括改变或结束该消息. 文献[11]介绍了利用 Hook 机制拦截和改变原有 API 执行的方法. BHO(Browser Helper Object) 是微软推出的浏览器对第三方程序员开放交互接口的标准. 程序员通过接口编写代码获取浏览器的行为、触发浏览器的行为, 以及执行特定处理.

SSO 客户端启动后, 以系统服务方式伺服于后台. SSO 客户端包含分别处理 C/S 应用和 B/S 应用的两个代理登录模块. C/S 中, 使用 Spy++或开发环境提取出窗口和组件信息, 对窗口激活事件进行 Hook. B/S 中, 当用户浏览器启动访问, 则加载 BHO 插件. 参考文献[12], 插件的实现思路包括: 登录页面网址识别、页面元素分析、网站登录账户和密码的自动填写.

用户运行 C/S 应用, 监测模块拦截到窗口事件, 判断是否为目标应用, 若不是则释放钩子; 若是, 则提取 Ticket 中的 KS, 解密 AuthenticatorC 中的应用认证, 填入相应组件, 激活提交. 若用户运行 IE 浏览器访问网站的时候, 系统 BHO 插件自动生成, 提取和匹配网址, 对 SSO 中 WEB 应用的登录界面分析元素并填入认证信息, 激活提交. 如图 3 所示:

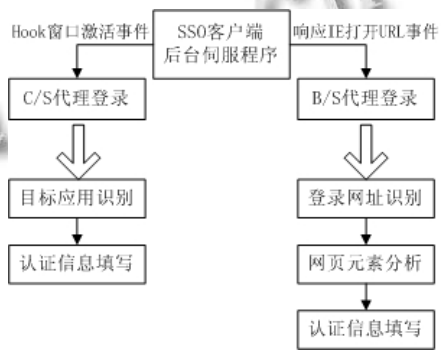

图 3 SSO 客户端后台伺服程序

3 结语

本文所涉及的认证模式已在广东省某市国税局统一门户系统中实施. 该系统所涉及业务应用包括了 B/S 和 C/S 体系结构, SSO 服务模块采用 C/S 架构模式, 认证服务器 AS 结合国税已有的 CA 认证服务体系实现. 比较采用 CAS 技术和采用本方案的单点登录实现方式: CAS 中票据存活期默认为 120 分钟, 即 2 小时内重发攻击有效, 而用户需要每隔 2 小时重新登录; 采用 checkin 模式, 用户只需一次登录, 一次主动注销或者被动注销认证身份, 重登时间不受限制, 重发攻击的可能发生期缩短为一次 checkin 周期, 在实现中, checkin 周期设置为 5 分钟, 大大降低了被攻击的风险.

但本系统的 SSO 客户端采用了 Hook 和 BHO 技术, 使应用局限于 Windows 环境, 虽然适应本项目及大部分的应用环境, 但在兼容多种客户端环境下仍需作进一步探索.

参考文献

1 Stallings W. Cryptography and Netork Securit:Principles and Pracices(5th Ed). Prentice Hall, 2010.

2 Milenkovic I, Latinovic O, Simi D. Using kerberos protocol

for single sign-on in identity management systems. Journal of Information Technology and Applications, 2013, 6: 27–33.

3 李小标,温巧燕,代战锋.PKI/PMI 支持多模式应用的单点登录方案.北京邮电大学学报,2009,32(3):104–108.

4 Armando A, Carbone R, Compagna L, Cuellar J, Pellegrino G, Sorniotti A. From multiple credentials to browser-based single sign-on. IFIP International Federation for Information Processing. SEC 2011. 68–79.

5 Zwattendorfer B, Tauber A. Secure cross-cloud single sign-on (SSO) using eIDs. Internet Technology and Secured Transactions, 2012 International Conferece for IEEE Conference Publications. 2012. 150–155.

6 孙从友,徐国胜,钟尚勤.一种基于公钥密码体制的 Kerberos 认证方案.第八届中国通信学会学术年会论文集,2011.

7 Li G. Security risk of depending on synchronized clocks. Operating Systems Review, 1992, 26(1): 49–53.

8 曹喆,王以刚.基于 USBKey 的身份认证机制的研究与实现.计算机应用与软件,2011,28(2):284–286.

9 陈东,尹建伟.支持多模式应用的分布式 SSO 系统设计与实现.计算机应用研究,2007,24(4):276–278.

10 徐江峰,邵向阳.基于 HOOK API 技术的进程监控系统设计与实现.计算机工程与设计,2011,32(4):1330–1333.

11 苏雪丽,袁丁.Windows 下两种 API 钩挂技术的研究与实现.计算机工程与设计,2011,32(7):2548–2552.

12 徐研,张伟.基于 BHO 的淘宝网账户自动登录系统研究与实现.软件导刊,2011,10(4):123–125.