

## 一个日志完整性检测方法

陈黎明<sup>1,2</sup>, 俞研<sup>1,2</sup>, 黄皓<sup>1,2</sup>

(1. 南京大学 计算机科学与技术系, 江苏 南京 210093;

2. 计算机软件新技术国家重点实验室, 江苏 南京 210093)

(mg0233003@ymail.nju.edu.cn)

**摘要:**通常入侵者在成功控制系统后会试图更改日志文件以消除入侵痕迹, 隐藏入侵行为。为了防止入侵者隐藏其入侵行为, 提出了一个日志完整性检测方法, 对日志的完整性进行检测, 使得入侵者不能不被发现地更改系统被其控制以前在日志文件中写入的记录, 进而提供保护。并在日志完整性受到破坏时, 给出一个可信任日志记录集合以供其他程序使用。

**关键词:**日志完整性; 单向哈希函数; 报文鉴别码

**中图分类号:** TP309 **文献标识码:** A

## Method for verifying log integrity

CHEN Li-ming<sup>1,2</sup>, YU Yan<sup>1,2</sup>, HUANG Hao<sup>1,2</sup>

(1. Department of Computer Science and Technology, Nanjing University, Nanjing Jiangsu 210093, China;

2. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing Jiangsu 210093, China)

**Abstract:** All kinds of system events are stored in logs. After a successful intrusion, the intruder will try to modify the logs to conceal the intrusion. A method for verifying and protecting log integrity was described to make all log entries generated prior to the logging system's compromise impossible for the attacker to undetectably modify. A set of trusted log entries was provided to other programs when damages are made to log integrity.

**Key words:** log integrity; one-way hash; MAC

## 0 引言

日志记录了系统中的各种事件, 包括入侵行为引发的事件。通过分析日志, 可以发现系统中的异常。通常入侵者在成功入侵一个系统后, 会试图消除其在系统中留下的痕迹, 通过删除或修改与其入侵行为相关的日志记录, 使其入侵行为不被发现。入侵者也可能在日志文件中恶意插入伪造的日志记录, 干扰系统的正常工作。

入侵者删除或修改日志记录会导致日志完整性被破坏, 日志完整性检测则可以让管理员及时知道系统已经被入侵者成功入侵, 从而采取相应措施, 比如检查、消除入侵者留下的后门等, 降低入侵行为造成的危害。

通过日志完整性检测可以判断日志信息是否可信任, 完整性未被破坏的日志信息是可信任的。现行的系统完整性检测工具一般不提供日志完整性检测功能, 为此我们提出了一个日志完整性检测方法。本文的讨论基于这样的设定: 攻击者成功利用系统漏洞入侵系统, 获取超级用户权限后就获得了对这个系统, 包括内核的完全控制权。入侵者能读写存储在系统中的任何数据, 包括日志信息, 能读取甚至修改内存中的数据。

## 1 方法介绍

系统配置文件以及可执行程序很少发生改动。对这些文件进行完整性检测相当直观。对每个文件的属性、内容等信息计算报文鉴别码, 并将计算报文鉴别码时使用的密钥和报文鉴别码一起妥善保存起来。验证文件的完整性时, 利用保

存的密钥和相应的哈希函数对文件重新计算报文鉴别码, 并与保存的报文鉴别码比较, 就可以判断文件的完整性是否被破坏。而日志文件则处于不断改动中, 新的日志记录不停被追加到日志文件中。因而对普通文件进行完整性检测的方法不能直接应用到日志文件上。

每条日志记录有两个与顺序相关的潜在属性:

(1) 序列号。每条日志记录在写入时都对应一个序列号, 最先写入日志文件中的日志记录序列号为0, 其次写入的为1。

(2) 位置。日志文件中的每条日志记录在日志文件中都有相应的位置。第一条日志记录位置为0, 第*i*条日志记录位置为*i*-1。

显然, 如果入侵者在日志文件中插入或者删除了日志记录, 则日志文件中插入或者删除位置之后的日志记录序列号与其位置不等。序列号和位置是日志记录的潜在属性, 日志记录中并没有日志记录序列号和位置这两个属性, 不能直接利用这两个属性对日志完整性进行检测。而且, 这两个属性不能检测入侵者只是修改日志记录, 而不删除或者插入日志记录的情形。所以, 我们仍需采用报文鉴别码技术。

在该方法中, 日志系统对每条日志记录计算报文鉴别码, 并将报文鉴别码附加在日志记录末尾写入日志文件。计算报文鉴别码时使用的密钥推导方法如下:

日志系统在开始记录日志之前产生一个随机数 $K_0$ , 作为对序列号为0的日志记录作MAC时使用的密钥。对序列号为*i*的日志记录作MAC时使用的密钥 $K_i$ 通过单向哈希函数*H*作用在对序列号为*i*-1的日志记录作MAC时使用的密钥 $K_{i-1}$

收稿日期: 2004-09-23; 修订日期: 2004-12-07

作者简介: 陈黎明(1981-), 男, 安徽池州人, 硕士研究生, 主要研究方向: 计算机网络安全; 俞研(1974-), 男, 吉林人, 博士研究生, 主要研究方向: 计算机网络安全; 黄皓(1957-), 男, 江苏海门人, 教授, 博士生导师, 主要研究方向: 网络与信息安全。

和序列号  $i$  上得到, 即  $K_i = H(K_{i-1}, i)$ 。得到  $K_i$  后,  $K_{i-1}$  在内存中销毁。日志系统只保留对当前日志记录作 MAC 需要的密钥, 以防止入侵者进入系统之后获得系统攻陷之前的密钥。如果入侵者获得了系统攻陷之前的密钥, 就可以在系统攻陷之前存储的日志记录中删除, 修改或者插入日志记录而不被发现。 $K_0$  在产生之后, 立即被存放到安全的地方, 供完整性检测时使用。

日志记录过程如图 1 所示,  $D_i$  表示序列号为  $i$  的日志记录,  $K_i$  表示对  $D_i$  作 MAC 时使用的密钥,  $A_i$  是利用  $K_i$  对  $D_i$  计算出来的报文鉴别码。

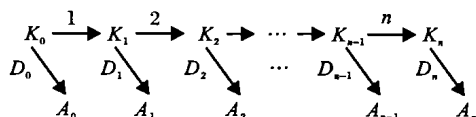


图1 日志记录过程

验证完整性是否被破坏时, 日志系统重新获得  $K_0$ , 模仿日志记录过程对日志文件中的每条日志记录 (不包括附加在日志记录尾部的报文鉴别码) 重新计算报文鉴别码。计算报文鉴别码时使用的密钥推导方法与日志记录过程使用的密钥推导方法类似。 $V_0 = K_0$ , 用来对位置 0 的日志记录计算报文鉴别码。位置为  $j$  的日志记录使用的密钥通过  $V_j = H(V_{j-1}, j)$  得到。如果对位置为  $j$  的日志记录计算出的报文鉴别码与附加在其末尾的报文鉴别码不等, 则日志文件完整性遭到破坏。位置 0 到位置  $j-1$  的日志记录可被信任。位置  $j$  到日志文件末尾的日志记录不被信任。

## 2 具体描述

### 2.1 日志系统模型

普通的日志系统可以划分为两个部分:

日志源: 被记录信息的来源, 就是系统中各种产生日志的程序;

日志记录模块: 用于记录日志源产生的日志数据, 就是一个在系统中负责存储日志信息的程序。

本方法引入第三个部分: 完整性检测模块。完整性检测模块负责对日志的完整性进行检测。日志记录模块从日志源获得日志记录, 对日志记录作完整性检测要求的处理 (计算日志记录的报文鉴别码) 后, 将日志记录存储到日志文件中。完整性检测模块从日志文件中读取日志记录, 进行完整性检测。本文方法中, 日志系统模型如图 2 所示。

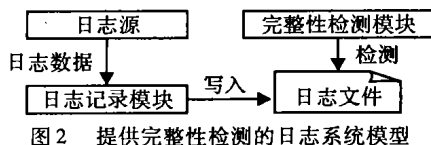


图2 提供完整性检测的日志系统模型

### 2.2 具体过程

完整性检测模块向日志系统提供了日志完整性检测功能。但日志记录模块在存储日志记录时, 必须对日志记录做相应处理, 完整性检测模块才能检测日志的完整性。这一节用伪码描述日志记录模块和完整性检测模块的工作。为了表达方便, 作定义如下:

$MAC(K, D)$  表示用密钥  $K$  对日志记录  $D$  作报文鉴别码;

$H$  表示单向哈希函数;

$\langle D, A \rangle$  表示日志记录  $D$  与其报文鉴别码  $A$  的连接;

$i$  表示当前日志记录序列号;

$j$  表示当前日志记录在日志文件中的位置;

$n$  表示日志文件中的日志记录数。

过程: 日志记录模块记录日志过程

输入: 日志源产生的日志记录

输出: 附加了报文鉴别码的日志记录

BEGIN

从完整性检测模块获得初始随机数  $K_0$ ;

$i = 0$ ;

WHILE (不需要检测日志完整性)

{ 从日志源获得  $D_i$ ;

$A_i = MAC(K_i, D_i)$ ;

$K_{i+1} = H(K_i, i + 1)$ ;

将  $\langle D_i, A_i \rangle$  写入日志文件;

$K_i$  在内存中销毁;

$i = i + 1$ ;

}

END

什么时候检测日志完整性根据需要而定。可以在记录日志数据一段时间后或者在日志文件中记录数目超过某一阈值时进行检测。管理员也可向日志记录模块发送信号, 告知需要检测日志完整性。

完整性检测模块负责两个工作: 产生初始随机数和检测日志完整性。

过程: 完整性检测模块产生初始随机数过程

输入: 日志记录模块的请求

输出: 初始随机数

Begin

接收日志记录模块要求初始随机数的请求;

产生初始随机数  $K_0$ ;

将  $K_0$  存放到安全的地方;

将  $K_0$  发送给日志记录模块;

END

过程: 完整性检测模块检测完整性过程

输入: 日志文件

输出: 日志完整性情况和可信任日志记录集合

BEGIN

获得存放在安全地方的初始随机数  $K_0$ ;

$V_0 = K_0$ ;

$j = 0$ ;

WHILE ( $j < n$ )

{ 从日志文件中读取  $\langle D_j, A_j \rangle$ ;

$A = MAC(V_j, D_j)$ ;

$V_{j+1} = H(V_j, j + 1)$ ;

IF ( $A$  不等于  $A_j$ )

{ 报告日志完整性被破坏;

报告  $D_j$  之前的记录可信任;

停止检测;

}

$j = j + 1$ ;

}

报告完整性未被破坏;

END

## 3 分析

### 3.1 检测和保护对象

假定入侵者在时间  $T$  入侵系统。系统中时间  $T$  之前产生的密钥 (包括初始随机数) 已经在内存中销毁, 而初始随机数存放在安全的地方, 不能被入侵者获得, 因而入侵者不能构造出时间  $T$  之前使用的密钥, 从而无法更改时间  $T$  之前写入的日志记录而不被发现。这个方法只能检测和保护时间  $T$  以前写入的日志记录的完整性是否被破坏, 不能检测和保护时间  $T$  后写入的日志记录的完整性。因为在系统被入侵者控制后, 入侵者可以从内存中读取入侵成功时刻日志记录模块正在使

用的密钥,按照前面提到的日志记录过程在日志文件末尾追加日志记录而不会被检测出来。但入侵者如果只在日志末尾追加日志记录,其入侵痕迹会在日志文件中保留下来,日志分析程序对日志文件进行分析时将会发现异常而告警。

### 3.2 初始随机数的安全

方法的安全性基于这样的设定:入侵者成功控制系统后,即使获得了当前的密钥,也不能反逆单向哈希函数产生系统被控制之前日志记录模块使用的密钥。只有知道初始随机数才能重新产生为每个日志记录作 MAC 时使用的密钥,从而改动日志记录而不被发现。所以初始的随机数必须妥善保存,否则安全性就无从谈起。

比较方便的方法是通过安全信道将初始随机数发送到另一台机器上存储。在验证完整性时,重新启动一个安全信道,将初始随机数传递给完整性检测模块。但存放初始随机数的系统也被入侵时,初始随机数就会被入侵者获得。更好的办法是采用 Tripwire 保护密钥的方法,完整性检测模块产生初始随机数后,将初始随机数写入软盘或者可擦写光盘。软盘或者可擦写光盘随即脱离系统,完整性检测模块在开始检测日志完整性时,提示用户将软盘或者可擦写光盘插入到系统中,读取初始随机数。后一种方法虽然麻烦,但要安全的多。

表1 性能数据

写入的日志记录数	普通日志系统	本方法
500	0.003s	0.020s
1000	0.003s	0.037s
2000	0.005s	0.072s
4000	0.008s	0.143s
8000	0.015	0.284s
16000	0.028s	0.566s
32000	0.055s	1.134s
64000	0.109s	2.262s
128000	0.220s	4.655s

### 3.3 性能

实验环境:CPU Intel P4 1.7G;操作系统 Red Hat 9.0;单向哈希函数和 MAC 均采用 SHA1 算法;从一个记录数约为 200000 的废弃日志文件中读取日志记录,发送给日志记录模

块作为日志源。 $n$  为日志记录模块写入到日志文件中记录数。

普通的日志系统日志记录过程如下:日志记录模块从日志源得到日志记录,将其转存到磁盘上(或通过网络转发给日志服务器)。本文方法与普通日志系统的日志记录过程比较,存储每条日志记录时都多了密钥推导和计算报文鉴别码的开销。表1对比了普通日志系统与本方法的日志记录过程的开销。可以看到本方法的日志记录过程计算开销比普通日志系统约高一个数量级。写入过程没有循环,时间复杂度为  $O(n)$ 。记录 100000 条日志记录耗时约 4s。

完整性检测过程开销:同写入过程一样,完整性检测过程没有循环,时间复杂度为  $O(n)$ 。检测 10000 条日志记录所用时间与日志记录过程耗用的时间相当。从实验数据看,本文方法计算开销并不大,一般的系统都可以接受,不会造成系统整体性能的大幅降低。

## 4 结语

本文给出了一个检测日志完整性的方法。通过该方法,入侵者不能更改在系统被其控制以前写入的日志记录而不被发现,使得入侵行为不可能长时间不被发现。在完整性受到破坏时,该方法会给出一个可信任的日志记录集合供日志分析程序或者其他需要利用日志数据挖掘有用信息的应用程序使用。该方法的开销不大,不会对系统性能造成明显影响。

### 参考文献:

- [1] BELLARE M, YEE BS. Forwarded integrity for secure audit logs [Z]. 1997.
- [2] SCHNEIER B, KELSEY J. Cryptographic Support for Secure Logs on Untrusted Machines [A]. Proceedings of the 7th USENIX Security Symposium [C]. 1998.
- [3] AYRATEPOV D, GANAPATHI A, LEUNG L. Improving the Protection of Logging Systems [DB/OL]. <http://www.cs.berkeley.edu/~archanag/publications/privacypaper.pdf>, 2004.
- [4] STALLINGS W. 密码编码学与网络安全 [M]. 北京:电子工业出版社, 2004.
- [5] DIFFIE W. 应用密码学 [M]. 北京:机械工业出版社, 2003.
- [6] FOUGERE J. An Introduction to Tripwire [EB/OL]. <http://www.webpronews.com/it/security/wpn-23-20020226AnIntroductiontoTripwire.html>, 2004.

(上接第 866 页)

一直以来,在企业信息系统的开发中,与访问控制相关的具体业务逻辑往往散落在系统的各个部分,不能做到集中管理,难以应对企业业务流程的变化,使得系统的维护工作量特别大。这里我们提出以 XML 的格式来描述企业的安全策略,并在运行时捕获上下文信息来进行访问控制的方案,它与传统的基于角色的访问控制的比较如表 1 所示。

## 3 结语

本文针对传统 RBAC 模型在应用中的不足,指出存在使能型权限、激活型权限和限制型权限这三种不同类型的权限,并且引入了上下文和规则的概念,提出了基于角色和规则的访问控制模型。这样安全管理员只要给权限指定关联的客体集范围,系统通过在运行时捕获上下文信息并应用安全策略来过滤和限制原始分配的资源集合,得到提交请求的用户能真正作用的客体集,从而大大降低了 RBAC 模型中角色权限分配的工作量。同样,系统在进行访问控制检查时,也能根据上下文信息来动态决定是否接受提交的请求,因而能提供更

细粒度的访问控制。在该模型的应用中,为了使系统能够实时响应提交的请求,如何更有效地进行安全策略的存储和搜索是关键,这将是下一步工作的重点。

### 参考文献:

- [1] SANDHU RS, COYNE EJ, FEINSTEIN HL, et al. Role-based access control models [J]. IEEE Computer, 1996, 29(2): 38-47.
- [2] 刘道斌, 白硕. 基于 workflow 状态的动态访问控制 [J]. 计算机研究与发展, 2003, 40(3): 417-421.
- [3] 袁平鹏, 陈刚, 董金祥. 多政策的两层协同应用存取控制模型 [J]. 计算机辅助设计与图形学学报, 2004, 16(4): 420-426.
- [4] MCPHERSON D, Microsoft Corporation. Role-Based Access Control for Multi-tier Applications Using Authorization Manager [EB/OL]. <http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/management/athmanwp.mspx>, 2004-07.
- [5] 李成错, 詹永照, 茅兵, 等. 基于角色的 CSCW 系统访问控制模型 [J]. 软件学报, 2000, 11(7): 931-937.
- [6] 韩伟力, 陈刚, 尹建伟. 权限约束支持的基于角色的约束访问控制模型与实现 [J]. 计算机辅助设计与图形学学报, 2002, 14(4): 333-339.