一种基于局部性原理的远程验证机制

夏庆勋 庄 毅
(南京航空航天大学计算机科学与技术学院 南京 211106)

摘 要 为了提高嵌入式平台配置远程证明方案的效率,在基于Merkle哈希树存储结构的基础上,结合程序的局部性原理,考虑平台下程序验证的时间特性,对存储程序模块完整性度量值的数据结构进行了改进,提出了一种基于局部性原理的远程验证机制。实验分析表明,新的机制可以减少构造存储度量日志的时间消耗,缩短应用程序实时认证路径的长度,提高平台配置远程证明的验证效率。

关键词 可信计算,远程证明,Merkle哈希树,局部性原理
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.04.024

Remote Attestation Mechanism Based on Locality Principle

XIA Qing-xun ZHUANG Yi
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

Abstract In order to improve the efficiency of the remote configuration attestation scheme, combining the locality principle of the program with the storage structure of Merkle Hash tree, the data structure used to store the Hash values of the program module integrity was improved, and a remote proof mechanism based on locality principle was proposed. Experiments show that the new mechanism can improve the efficiency of the remote configuration attestation by reducing the consumption of constructing stored measurement logs and shortening the length of authentication paths.

Keywords Trusted computing, Remote attestation, Merkle Hash tree, Locality principle

1 引言

随着人们对网络安全问题的日益关注,基于硬件可信根的可信计算已经发展成为保护计算基础设施和数十亿终端的工业技术[1]。可信计算小组(Trusted Computing Group, TCG)也正在将可信计算的概念由传统的带有可信平台模块(Trusted Platform Module, TPM)的PC机扩展到其他设备。可信计算技术现已在企业网络、嵌入式系统、移动设备等领域被广泛应用[2-4]。

可信计算技术可以通过强制系统执行特定的行为来保护其免遭未经授权的更改和攻击,使得根工具包(rootkit)等恶意软件无法对其造成攻击和破坏[5]。通过这一技术,系统、网络和应用变得更加安全,不仅不易受病毒和恶意软件的影响,而且更易部署和管理。

远程证明(Remote Attestation, RA)是可信计算平台的一个基本特征[1],指当某一终端节点(证明方)向远程的可信节点(质询方)请求服务时,通过特定的步骤,由远程的可信节点对请求服务的节点进行验证,判断其是否处于可信的状态,从而决定是否为其提供服务。通常情况下,质询方需要对证明方进行平台身份验证和平台配置完整性验证[6]。

在对平台配置进行完整性验证的过程中,需要将完整性度量值存储在存储度量日志中。本文将局部性原理的概念引入到存储度量日志的构建中,提出基于局部性原理的远程验证机制,该机制可提高存储度量日志的构建效率和证明方的验证效率。

本文第2节首先介绍平台配置远程证明的相关研究工作;第3节介绍局部性原理,并在此基础上介绍LPBHT的构造机制和平台完整性验证过程;第4节对提出的远程验证机制进行效率分析与评估;最后总结全文。

2 相关研究工作

平台配置远程证明是远程证明中的一个重要过程。TCG规范提出的平台完整性证明架构如图1所示。

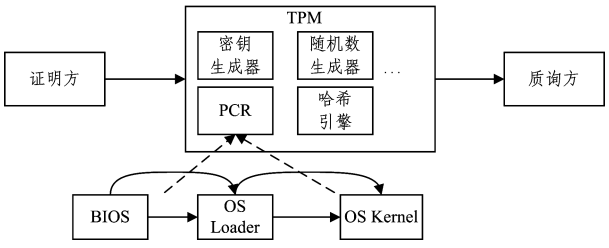


图1 TCG规范完整性证明架构

Fig. 1 TCG integrity attestation architecture

该架构基于信任传递模型(Transitive Trust Model, TTM),证明方在系统启动的过程中代码运行前逐层进行完整性度量,并将度量的哈希值存放到TPM的平台配置寄存

到稿日期:2017-01-07 返修日期:2017-03-13 本文受国家自然科学基金(61572253),航空科学基金(2016ZC52030)资助。
夏庆勋(1992-),男,硕士生,主要研究方向为网络安全、可信计算;庄 毅(1956-),女,教授,主要研究方向为网络安全、可信计算、分布并行计算,E-mail:zy16@nuaa.edu.cn(通信作者)。

器(Platform Configuration Register，PCR)中。PCR 由于只可以进行重置操作(reset)和扩展操作(extend)，因此具有防篡改的安全特性，从而使得该架构可以如实地反映系统从 BIOS 到操作系统的完整性状态[7]。

PCR 的扩展操作可由式(1)表示：

$$PCR_i^{t+1}=Hash_{alg}(PCR_i^t /\!/ x) \tag{1}$$

其中，$Hash_{alg}$表示所采取的哈希算法，x 表示需要扩展的内容，// 表示连接操作。由于该机制只度量到操作系统层，忽略了应用程序的可信性，Sailer 和 Jaeger 等人[8-9]对其进行了扩展，提出了 IMA，PRIMA 等架构。TCG 也在 TPM2.0 规范中加入了应用程序的完整性验证[10]。但由于需要对所有运行的程序模块进行度量与验证，它们普遍存在隐私泄露和效率较低等问题。

徐梓耀等人利用 Merkle 哈希树，提出了一个保护隐私的证明架构 RAMT(Remote Attestation Mechanism based on Merkle Hash Tree)[11]。其利用 Merkle 哈希树中的叶子节点来存储模块度量的哈希值，利用非叶子节点存储其两个孩子节点的串联哈希值，因此根节点包含了所有模块的完整性信息，只需要保证根节点的哈希值不被篡改，就可以保证全部叶子节点的完整性。Merkle 哈希树的结构如图 2[11]所示。该方法可在一定程度上避免证明方的隐私泄露问题，并提高了验证的效率；但该方法由于未能反映应用程序实时运行的问题，不能对证明方的平台配置进行动态度量。

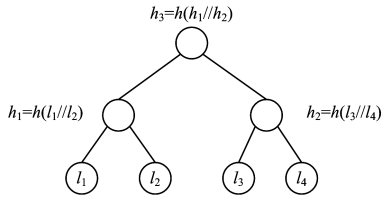


图 2　Merkle 哈希树的结构示意图

Fig. 2　Structure of Merkle Hash tree

朱毅等人[12]考虑到哈希树的构建效率问题，提出了一种基于左满树的完整性度量模型。该模型虽然对哈希树的构建进行了优化，减少了哈希树构建的时间开销和空间开销，但未能提高远程证明的验证效率。Fu 等人[13]考虑到应用程序的访问频率问题，提出了一种非平衡树的存储度量日志(Unbalance Tree Structure Storage Measurement Log，UBTS-SML)构造方法。该方法虽然提高了远程证明的验证效率，但是需要针对每一个应用程序的请求更新树的结构，哈希树的更新维护效率较低，并且没有考虑到叶子节点的撤销操作。

为了进一步提升平台配置远程证明中哈希树的更新维护效率和远程证明的验证效率，本文提出了基于局部性原理哈希树(Locality Principle-Based Hash Tree，LPBHT)的远程验证机制。该机制通过引入时间局部性的概念，并考虑程序度量值在存储度量日志中的撤销问题，使得哈希树的更新效率和远程验证效率都有了较大的提升。

3　基于局部性原理的远程验证机制

3.1　局部性强弱的量化方法

局部性原理是指研究人员在对程序进行分析时，发现数据的访问是阶段性的，并表现出一种聚集的现象[14]。这种聚集通常体现在两个维度：时间的局部性和空间的局部性。时间的局部性是指一个数据被访问之后，在不久后很可能会被再次访问；空间的局部性是指一个数据被访问之后，地址相邻的数据在不久后也可能会被访问。

在远程证明的完整性度量阶段，需要将程序的哈希值存放在存储度量日志中。由于程序的动态性，通常需要增加和更新存储度量日志，时间的局部性就体现在当某一程序因请求服务而被更新到存储度量日志中后，很可能在不久之后会再一次请求服务。利用这一特性构造存储度量日志，会使得远程证明的验证效率得以提升。为了量化时间的局部性强弱，给出程序请求服务情形下时间的局部性强度的定义。

定义 1(时间的局部性强度)　设在时间间隔 $t_1,t_2,\cdots,t_n$ 下，请求服务的程序集合为 $s_1,s_2,\cdots,s_n$，IOS_i 表示 t_i 间隔与 t_{i+1}间隔请求服务的程序集合的交集，UOS_i 表示 t_i 间隔与 t_{i+1}间隔请求服务的程序集合的并集，即：

$$\begin{aligned}&IOS_1=s_1\cap s_2,UOS_1=s_1\cup s_2\\&IOS_2=s_2\cap s_3,UOS_2=s_2\cup s_3\\&\cdots\\&IOS_{n-1}=s_{n-1}\cap s_n,UOS_{n-1}=s_{n-1}\cup s_n\end{aligned} \tag{2}$$

设 $card()$表示集合中元素的个数，则时间的局部性强度 IOL(Intensity of Locality)可由式(3)定义：

$$IOL_i=card(IOS_i)/card(UOS_i) \tag{3}$$

定理 1　IOL_i 的取值范围为[0，1]。

证明：$IOL_i=card(IOS_i)/card(UOS_i)=card(s_i\cap s_{i+1})/card(s_i\cup s_{i+1})$，$0\leqslant card(s_i\cap s_{i+1})/card(s_i\cup s_{i+1})\leqslant card(s_i)/card(s_i)=1$。命题得证。

定理 2　IOL_i 的值与 t_{i+1}时刻再次请求服务的程序个数正相关。

证明：假设 $card(s_i)$，$card(s_{i+1})$为定值，则

$$\begin{aligned}IOL_i&=\frac{card(IOS_i)}{card(UOS_i)}=\frac{card(s_i\cap s_{i+1})}{card(s_i\cup s_{i+1})}\\&=\frac{card(s_i\cap s_{i+1})}{card(s_i)+card(s_{i+1})-card(s_i\cap s_{i+1})}\\&=\frac{1}{\dfrac{card(s_i)+card(s_{i+1})}{card(s_i\cap s_{i+1})}-1}\end{aligned}$$

当 $card(s_i\cap s_{i+1})>card(s_i\cap s_{i+1})'$时，$IOL_i>IOL_i'$。

命题得证。

通过以上定理可知，当再次请求服务的程序数量越多时，IOL_i 的值越接近 1，时间的局部性表现得越强，相应地，基于局部性原理哈希树的远程验证机制的验证性能就越好。对于这一部分的论证，将在 4.2 节进行更详尽的说明。

3.2　LPBHT 的构造机制

LPBHT 使用一个受硬件保护的寄存器来存储程序度量哈希树的根结点，因而认为任何的非物理攻击(包括恶意软件、木马病毒等)都无法更改硬件保护寄存器的值；同时使用一个易失性存储器充当缓冲池来缓冲应用程序的请求，从而避免构造更新的操作过于频繁，增加构造机制的伸缩性，提高系统的性能。LPBHT 涉及到的操作主要包括程序最近访问频率的统计、哈希树的遍历、程序度量值的撤销以及哈希树的更新等。LPBHT 中结点的组成可由一个六元组表示：$\langle id,hvalue,rvf,lchild,rchild,parent\rangle$。其中，$id$ 为结点标识，每个应用程序具有唯一的结点标识；$hvalue$ 为程序的度量哈希值；rvf 为近期访问频率；$lchild$，$rchild$，$parent$ 为该结点的

左、右孩子结点和父结点。每当 LPBHT 进行更新之后，树的根结点(包含所有应用程序的完整性度量的串联哈希值)都需要被扩展到受硬件保护的寄存器中。

定义 2(HuffMHT) 设二叉树的叶子结点 W_1，W_2，…，W_n 存放程序的哈希值 $hvalue_1$，$hvalue_2$，…，$hvalue_n$ 和权值 $freq_1$，$freq_2$，…，$freq_n$，内部结点存放其子结点的连接哈希值 $H_{alg}(hvalue(lchild)||hvalue(rchlid))$和子结点的权值和，其中 H_{alg}是哈希算法，如 SHA1，SHA256，SM3 等，则 HuffMHT 是一棵根结点包含所有子结点信息并且带权路径和最短的二叉树。

假设一段时间间隔内共有 8 个应用程序请求 s_1，…，s_8，其权值为 0.05，0.06，0.07，0.08，0.09，0.15，0.17，0.33，则相应的 HuffMHT 如图 3 所示。

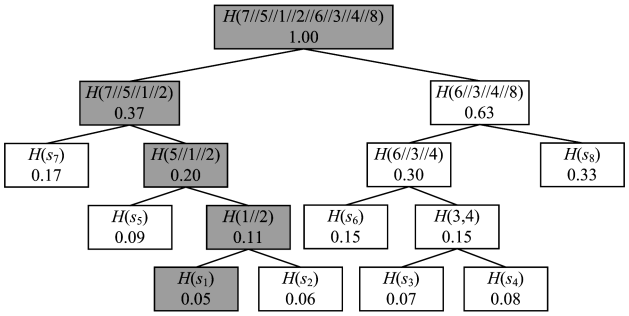


图 3 HuffMHT 示意图

Fig. 3 Schematic diagram of HuffMHT

算法 1 HuffMHT 的构造

步骤 1 生成森林 W，森林中的每棵树 W_0，W_1，…，W_n 都是一棵只有一个根结点的树，W=｛W_0，W_1，…，W_n｝。

步骤 2 生成一个根结点，在森林 W 中选出两个根结点的权值最小的树作为该结点的左、右子树，新的根结点权值为其左、右子树根结点权值之和，结点度量哈希值 hvalue 为两棵子树的串联哈希值。

步骤 3 从森林 W 中删除选取的两棵树，并将新树加入森林 W。

步骤 4 判断森林 W 中是否只剩一棵树，若不是，则执行步骤 2；否则，该森林 W 中的树即是生成的 HuffMHT，返回树的根结点 root。

定义 3(认证路径) 在进行平台完整性验证的过程中，证明方需要将哈希树中的一系列哈希值发送给质询方，使得质询方可以重新计算根哈希值以判断其正确性。认证路径是指哈希树中待验证的叶子结点到根结点路径上所有结点的集合。例如，图 3 中程序 s_1 的认证路径由灰色部分标出，其长度为 5。

定义 4(结点的撤销) 程序的完整性度量哈希值是存储度量日志中的核心，然而在实际应用中，即使是受信任的程序，也可能会产生含有安全漏洞的代码；此外，许多程序可能因生命周期已经结束而被用户卸载，不再使用。这些程序的完整性度量哈希值将不再被用于认证，因而成为无效的叶子结点。大量无效的叶子结点会在很大程度上影响验证的总体性能，因而需要将其从树中删除，这一操作定义为结点的撤销。

算法 2 结点的撤销

步骤 1 判断待撤销的结点是否为根结点 root 的孩子结点，若是，则执行步骤 2，否则执行步骤 3。

步骤 2 将待删除结点的兄弟结点置为根结点，并将其哈希值扩展保存到受硬件保护的寄存器中，释放撤销结点及原根结点的内存空间。

步骤 3 新建一个结点指针作为当前结点指向待撤销结点的父结点。

步骤 4 判断待撤销结点是当前结点的左孩子还是右孩子结点，并判断当前结点是其父结点的左孩子还是右孩子结点。

步骤 5 将当前结点的左(右)孩子指针指向当前结点父结点的左(右)孩子结点。

步骤 6 释放撤销结点及其父结点的空间。

步骤 7 重新计算当前结点的父结点的所有祖先结点的哈希值和权值，并将重新计算的根结点的哈希值保存到受硬件保护的寄存器中。

定义 5(近期访问频率，Recently Visited Frequency，RVF) 设在最近的一段时间间隔 t 内共有 m 个不同的程序请求访问，每个程序请求访问的次数为 r_i，则近期访问频率 RVF 可定义如下：

$$RVF=r_i/\sum_{i=1}^{m}r_i \tag{4}$$

由程序访问的局部性原理可知，若某一个程序在近期请求了访问，则在不久的一段时间内，该程序有很大的概率再次请求访问。相应地，近期访问频率更高的应用程序会有更高的概率再一次请求访问，对此进行如下形式化描述。

设程序 s_1 和 s_2 在一段时间间隔 t_i 的近期访问频率为 RVF_1 和 RVF_2，下一段时间间隔 t_{i+1} 的近期访问频率为 RVF_1'和RVF_2'，$P()$表示事件发生概率，则：

$$\exists\varepsilon>0,\ \text{s.t. } P((RVF_1>RVF_2\wedge RVF_1'>RVF_2')\vee(RVF_1<=RVF_2\wedge RVF_1'<=RVF_2'))>\varepsilon$$

基于以上假设，LPBHT 的核心机制就是使得近期访问频率更高的应用拥有更短的认证路径，从而节省远程验证过程的时间开销，提升其验证效率。LPBHT 的构造算法如算法 3 所示。

算法 3 LPBHT 的构造

步骤 1 判断 LPBHT 是否已存在，若不存在，则进行初始化，令 $id=-1$，$rvf=0$，生成一个根结点。

步骤 2 统计请求缓冲池中的应用程序的近期访问频率。

步骤 3 为所有请求缓冲池中的应用程序生成对应的叶子结点 W_1，W_2，…，W_n。

步骤 4 遍历 LPBHT，检测新生成的叶子结点标识号是否已经存在于哈希树中，若存在，则先执行算法 2；否则直接执行步骤 5。

步骤 5 将 LPBHT 的根结点看作 W_0 并将其加入 W 中。

步骤 6 利用 W_0，W_1，…，W_n 这 n+1 个结点来执行算法 1，即完成了 LPBHT 的构造更新。

步骤 7 将更新后的 LPBHT 的根哈希值通过扩展操作交由硬件保护的寄存器保存，清空缓冲池中的应用程序请求并接收新的应用程序请求。

3.3 基于 LPBHT 的完整性验证机制

当证明方的一个应用程序向质询方请求服务时，质询方需要对证明方进行平台的完整性验证和交互的远程证明。完整性验证机制如算法 4 所示。

算法 4 基于 LPBHT 的完整性验证机制

步骤 1 质询方生成一个随机数 nonce，并将 nonce 及要获取的 PCR 编号发送给证明方。

步骤 2 证明方装载身份证明密钥 AIK，将指定的 PCR 与 nonce 串联，并用 TPM 对其进行签名。

步骤 3 读取 LPBHT 的根哈希与 nonce 串联，并用 TPM 对其进行签名。

步骤 4　找到请求服务程序的认证路径。

步骤 5　将认证路径、签名的根哈希、PCR 发送给质询方。

步骤 6　质询方检查 nonce，验证 AIK 签名、PCR 的完整性和正确性。

步骤 7　质询方利用认证路径重新计算根哈希并进行比对，如果一致则验证通过。

4　分析与评估

4.1　构造效率

为了评估本文提出的 LPBHT 的构造效率，记录了不同叶子结点尺寸下构造存储度量日志所消耗的时间，并与平衡存储日志 MHT、Fu 等人提出的非平衡存储度量日志(UBTS-SML)进行对比。具体的实验环境是一台联想启天 M4390 微型计算机，计算机的配置为 Intel(R) Core(TM) i5-3470 CPU @3.20 GHz，4 GB 内存，系统为 ubuntu 15.04，64 位。利用 matlab R2013a 绘制出实验结果图，并与 UBTS-SML 和 MHT 的实验结果进行对比，结果如图 4 所示。

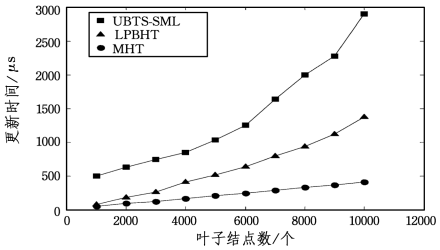


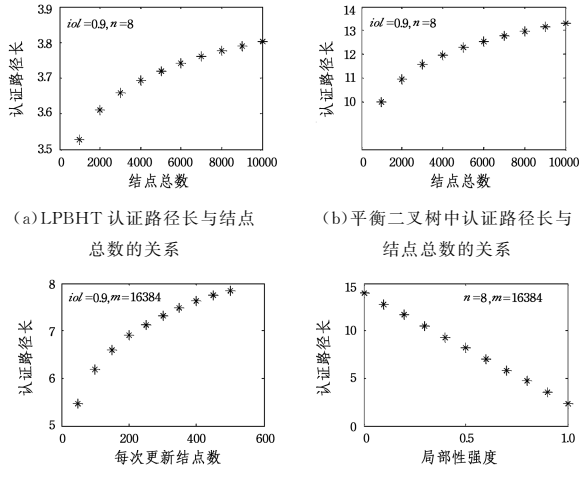
图 4　构造效率的对比

Fig. 4　Comparison of structural efficiency

实验结果表明，随着叶子结点个数的增加，构造存储日志所消耗的时间也会增加；LPBHT 的构造效率要明显高于 UBTS-SML；虽然 LPBHT 的构造效率低于 MHT，但是 MHT 没有考虑程序的动态性，属于静态度量方法，而 LPBHT 由于利用了局部性原理，验证效率要明显优于 MHT。

4.2　验证效率

为了评估 LPBHT 的验证效率，在 4.1 节所述的软硬件环境下，采用蒙特卡洛方法生成程序请求序列。记录平均认证路径长 *length_mean* 与局部性强度 *iol*、结点总数 *m*、每次更新结点数 *n* 的关系，如图 5 所示。



(a)LPBHT 认证路径长与结点总数的关系

(b)平衡二叉树中认证路径长与结点总数的关系

(c)LPBHT 认证路径长与每次更新结点数的关系

(d)LPBHT 认证路径长与局部性强度的关系

图 5　验证效率的关系

Fig. 5　Relationship of attestation efficiency

图 5(a)表示 LPBHT 中认证路径长与结点总数的关系，平均认证路径长在 3.5～3.9 之间变化。图 5(b)表示平衡二叉树中认证路径长与结点总数的关系，平均认证路径长在 10～14 之间变化。相比于平衡二叉树，LPBHT 的平均认证路径长缩短了大约 69.63%。

定理 3　程序的认证路径长度的期望与程序的局部性强度负相关。

证明：设 n 表示 LPBHT 在某一时间间隔 t 下更新的叶子结点数，m 表示原哈希树中的叶子结点数，iol 表示局部性强度。考虑最坏情况，即 LPBHT 结构下根结点的两棵子树是深度不同的两棵平衡二叉树，则 LPBHT 结构认证路径长度的期望可以表示为：

$$E(LPBHT)=iol\log n+(1-iol)\log m$$

$$=iol\log\frac{n}{m}+\log m \text{ 是关于 } iol \text{ 的一次函数}$$

因为 $n<m$

所以 $\log\frac{n}{m}<0$

命题得证。

定理 4　LPBHT 的认证路径长度期望小于平衡哈希树的认证路径长度期望。

证明：由定理 3 可得，LPBHT 结构认证路径长度的期望可以表示为：

$E(LPBHT)=iol\log n+(1-iol)\log m$

平衡哈希树结构的认证路径长度的期望可以表示为

$E(MHT)=\log(m+n)$

由于 $m>n$，不妨设 $m=kn$，其中 $k>1$，则

$$f(iol)=iol\log n+(1-iol)\log m-\log(n+m)$$

$$=iol\log n+(1-iol)\log kn-\log(k+1)n$$

$$=\log k-\log(k+1)-iol\log k$$

$$=\log\frac{k}{(k+1)k^{iol}}$$

$$=-\log(1+k)k^{iol-1}$$

$$=-(\log(1+\frac{1}{k})+iol\log k)<-iol\ \log k<0$$

因此 $iol\log n+(1-iol)\log m<\log(n+m)$

即 $E(LPBHT)<E(MHT)$。

由以上分析可知，当程序请求访问时，LPBHT 的认证路径要明显少于平衡结构哈希树的认证路径，当时间局部性强度越大时，性能表现越好，从而减小了质询方的运算负载，提升了验证的效率，并且随着程序结点规模的增大，这一特征将体现得更加明显。

结束语　本文提出了远程证明中的一种基于局部性原理的完整性度量验证机制，对 LPBHT 的构造进行了详细的描述与分析，并且提出了生命周期已经结束的程序完整性度量值在该机制中撤销的算法。实验表明，在保护隐私的安全强度不变的前提下，LPBHT 的构造具有较高的效率，并且该机制明显提升了远程证明的验证效率。

参 考 文 献

[1]　Trusted Computing Group(TCG)[OL]. http://www.trustedcomputinggroup.org.

(下转第 162 页)

有时不能很准确地描述分类器的目标函数。例如,在新类型的威胁指令刚出现时,其对应的分裂特征不大可能出现在决策树的上层,因为具有这种分裂特征的实例数量还比较少,所以其 EF 值也会较小。同理,对于新出现的分裂特征,其在其他决策树中出现的概率会偏小,导致 FF 值也较小。造成这种现象的原因是,在训练初期,权值容易受到分类器之间的相互干扰。因此,本文提出的算法在应对新攻击时还有问题需要解决,后续工作可以在这方面进行尝试和努力。

参 考 文 献

[1] PRADHAN B. A comparative study on the predictive ability of the decision tree, support vector machine and neuro-fuzzy models in landslide susceptibility mapping using GIS[J]. Computers & Geosciences, 2013, 51(2): 350-365.

[2] SHOTTON J. Real-time human pose recognition in parts from single depth images[J]. Communications of the ACM, 2013, 56(1): 116-124.

[3] FREUND Y, SCHAPIRE R E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting[J]. Journal of Computer and System Sciences, 1997, 55(1): 119-139.

[4] GLIGOROV V V, WILLIAMS M. Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree[J]. Journal of Instrumentation, 2012, 8(2): 6.

[5] RUTKOWSKI L. Trees for Mining Data Streams Based on the McDiarmid's Bound[J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(6): 1272-1279.

[6] SCHAPIRE R E. The strength of weak learn ability[J]. Machine Learning, 1990, 5(2): 197-227.

[7] XIE J B. Research and Implementation of Intranet Security Situation Awareness Technology[D]. Guangzhou: Guangdong University of Technology, 2015. (in Chinese)
谢锦彪. 内网安全态势感知技术的研究与实现[D]. 广州: 广东工业大学, 2015.

[8] BREIMAN L. Bagging predictors [J]. Machine Learning, 1996, 24(2): 123-140.

[9] SCHAPIRE R E. A brief introduction to boosting[C]// International Joint Conference on Artificial Intelligence. Sweden, 1999: 1401-1406.

[10] SCHAPIRE R E, SINGER Y. Improved boosting algorithms using confidence-rated predictions[J]. Machine Learning, 1999, 37(3): 297-336.

[11] WITTEN I H, FRANK E, HALL M A. Data Mining: Practical Machine Learning Tools and Techniques(Second Edition)[M]. San Francisco: Morgan Kaufmann publications, 2005.

[12] PAIK J H. A novel TF-IDF weighting scheme for effective ranking[C]// 36th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2013: 343-352.

[13] WU H C, LUK R W P, WONG K F, et al. Interpreting TF-IDF term weights as making relevance decisions[J]. ACM Transactions on Information Systems, 2008, 26(3): 55-59.

[14] ESCALANTE H J. Term-weighting learning via genetic Programming for text classification[J]. Knowledge-Based Systems, 2014, 83(1): 176-189.

[15] KUNCORO B A, ISWANTO B H. TF-IDF method in ranking keywords of Instagram users' image captions[C]// International Conference on Information Technology Systems & Innovation. 2015: 1-5.

(上接第 151 页)

[2] ASOKAN N, EKBERG J E, KOSTIAINEN K, et al. Mobile Trusted Computing[J]. Proceedings of the IEEE, 2014, 102(8): 1189-1206.

[3] FUGINI M G, BREVEGLIERI L, PELOSI G, et al. Trusted Computing for Embedded Systems[OL]. http://rd.spring.com/content/pdf/bfm%3A978-3-319-09420-5%2F1.pdf.

[4] MU Y. Zhong Guan Cun Trusted Computing Industry Alliance was Established[J]. Information Security and Communications Privacy, 2014(5): 16. (in Chinese)
木易. 中关村可信计算产业联盟成立[J]. 信息安全与通信保密, 2014(5): 16.

[5] SONG X L, ZHANG L H, CHEN D Y. Preventing Hypervisor-based Rootkit with Trusted Execution Technology[J]. Information Security & Communications Privacy, 2009, 7: 76-81.

[6] YU A, ZHAO S. Enhancing Flexibility of TCG's TNC through Layered Property Attestation[C]// IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE Computer Society, 2011: 751-756.

[7] ARTHUR W, CHALLENER D, GOLDMAN K. Platform Configuration Registers[M]// A Practical Guide to TPM 2.0. Apress, , Berkeley, CA, 2015.

[8] SAILER R, ZHANG X, JAEGER T, et al. Design and implementation of a TCG-based integrity measurement architecture [C]// Usenix Security Symposium. San Diego, CA, USA, 2004: 16-16.

[9] JAEGER T, SAILER R, SHANKAR U. PRIMA: policy-reduced integrity measurement architecture[C]// SACMAT 2006, ACM Symposium on Access Control MODELS and Technologies. Lake Tahoe, California, USA, 2006: 19-28.

[10] CAMENISCH J, CHEN L, DrRJVERS M, et al. One TPM to Bind Them All: Fixing TPM 2.0 for Provably Secure Anonymous Attestation[C]// Security and Privacy. IEEE, 2017: 901-920.

[11] XU Z Y, HE Y P, DENG L L. Efficient Remote Attestation Mechanism with Privacy Protection[J]. Journal of Software, 2011, 22(2): 339-352. (in Chinese)
徐梓耀, 贺也平, 邓灵莉. 一种保护隐私的高效远程验证机制[J]. 软件学报, 2011, 22(2): 339-352.

[12] ZHU Y, LI Q B, ZHONG C L, et al. Non-balanced Binary Hash-tree Model for Fine-grained Integrity Measurement[J]. Journal of Chinese Computer Systems, 2014, 35(7): 1604-1609. (in Chinese)
朱毅, 李清宝, 钟春丽, 等. 用于细粒度完整性度量的非平衡二叉哈希树模型[J]. 小型微型计算机系统, 2014, 35(7): 1604-1609.

[13] FU D, PENG X, YANG Y. Unbalanced tree-formed verification data for trusted platforms[J]. Security & Communication Networks, 2016, 9(7): 622-633.

[14] DENNING P J. The Locality Principle[J]. Communications of the Acm, 2005, 48(7): 19-24.