第39卷第11期 中国科学技术大学学报 Vol. 39, No. 11
2009年11月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Nov. 2009
文章编号:0253-2778(2009)11-1232-05

基于 Bloom filter 引擎的分布式网络取证系统

赵 骞,崔益民,邹 涛

(北京系统工程研究所,北京 100101)

摘要:针对目前网络取证的特点和技术挑战,提出和设计了一种基于 Bloom filter 引擎的分布式网络取证系统. 该系统以 Bloom filter 引擎为核心,能够实时的对网络原始数据进行过滤,映射压缩和存储,捕获完整的证据,节省存储空间,有效支持网络取证的事后分析查询.

关键词:网络安全;网络取证;Bloom filter;取证分析

中图分类号:TP393.08 **文献标识码**:A

Distributed network forensics system based on Bloom filter engine

ZHAO Qian, CUI Yi-min, ZOU Tao

(*Beijing Institute of System Engineering*, *Beijing* 100101, *China*)

Abstract: Aiming at technical challenges of network forensics at present, a distributed network forensics system based on the Bloom filter engine was proposed and designed. The system with the Bloom filter engine as the core can filtrate, memory-map raw network data for compression, capture complete evidence, save storage space, and help with post-event investigation of network forensics.

Key words: network security; network forensics; Bloom filter; forensics analysis

0 引言

网络取证主要是通过对网络数据流、主机系统日志等的实时监控和分析,发现网络系统中的入侵行为,自动记录犯罪证据,确定或获取潜在的、有法律效力的证据[1]. 网络取证事后分析机制是指在网络安全事件发生后很长一段时间才提取网络证据,重构网络安全事件,追索攻击者,为受害者提供法律证据. 网络取证事后分析机制为网络安全提供了新型的防护机制,不依赖于现有攻击特征和手段.

早期的网络取证技术和取证产品均侧重于网络入侵的追踪发现,利用 IDS, HoneyPot 日志等作为分析数据,但日志信息缺乏重构网络事件充足的数据,并不能实现网络取证的事后分析功能. 现在的网络取证产品,如美国 Network Associates 公司的 InfiniStream Security Forensics, Raytheon 公司的 SilentRunner, Niksun 公司的 NetDetector,都是采集原始网络数据,并提供强大的分析和数据显示功能重构网络事件. 但是巨大的网络数据流量和有限的数据存储之间的矛盾一直无法解决,作为事后分析的证据数据,要求系统能够保留数月甚至数年,巨大的存储成本开销使得这些取证系统不能大规模应用. 解决的办法就是对证据数据进行大比例压缩,并且不影响取证分析效果,为此本文提出一个以 Bloom filter(BF)引擎为核心的分布式网络取证系统,该系统通过 Bloom filter 数据结构对网络数据包进行映射压缩,从而大大降低网络数据的存储需求,使网络数据包能够在有限存储空间下,大范围采

收稿日期:2008-06-30;**修回日期**:2008-11-25

作者简介:赵骞,男,1982 年生,硕士生. 研究方向:网络安全. E-mail:fox086@sina.com

通讯作者:崔益民,副研究员. E-mail:tsui-min@163.com

集证据数据并长时间存储,提供事后查询所需的完整证据数据.

1 Bloom filter[4~6]

BF 是一个非常优秀的数据结构,由 Bloom 在 1970 年提出,现已广泛应用于计算机系统中庞大的数据集的表达,用于提高查找效率.

BF 的基本原理是:分配一个具有 m 位的向量 $\boldsymbol{V}$,所有的位被初始化为 0. 选择 k 个相互独立的 Hash 函数 $h_1,h_2,h_3,\cdots,h_k$,值域为$\{1,2,\cdots,m\}$. 对于 n 个数据对象的集合 $S=\{s_1,s_2,\cdots,s_n\}(m>n)$的每个数据对象计算一个地址序列 $h_1(s),h_2(s),\cdots,h_k(s)$,然后将位向量对应地址序列位置 1. 查询某个数据对象是否是集合 S 的成员时,需要检查表示 S 的 BF 对应数据对象地址序列的位,如果均为 1,则判定该数据对象属于 S,否则不属于 S. 在 BF 算法中可能对位向量中的同一个位多次置 1,所以在查询数据对象时可能存在“假肯定”(false positive)现象.

假肯定率(FP)的计算公式为

$$FP=(1-(1-1/m)^{kn})^k\approx(1-e^{-kn/m})^k \quad (1)$$

对式(1)进行求导,可推出 当 $k=\ln 2*(m/n)$,FP 取得最小值

$$FP_{min}\approx 0.618\ 5^{m/n} \quad (2)$$

由式(1)变换得

$$m=-\frac{nk}{\ln(1-FP^{1/k})} \quad (3)$$

在已知 n,FP 的情况下,可以证明存在一个 k 值,使得 m 取得最小值且唯一. 对式(3)进行求导,解得

$$k=-\log_2 FP \quad (4)$$

此时 m 最小值为

$$m=-\frac{n*\log_2 FP}{\ln 2} \quad (5)$$

定义存储空间压缩率

$$d=1-\frac{m}{n*b} \quad (6)$$

式中,b 为块的尺寸. 固定 m/n,减小 k,可以获得 70%以上的压缩率,大大减少数据存储量,牺牲的是 FP. 同样,也可以通过增大 m,减小 k 和 FP 的方法来获得较大的压缩率.

2 基于 Bloom filter 引擎的分布式网络取证系统

BF 通过小概率的假肯定换来了空间效率,将其应用于分布式网络取证,将大大降低数据证据的存储量和通信量. 合理配置 BF 参数,不仅能够保证证据查询的有效性,还能够较好地增加集合元素的时间复杂度、查找算法复杂度 $O(k)$以及压缩率 d[6]. 在查询单个 BF 时候,仅仅和参数 m 有关,与所映射存储的数据元素个数没有关系,这种特性使得 BF 能更加高效地应用于取证查询.

系统将以 BF 引擎作为核心部件,采集网络原始数据包映射存储,提取网络安全事件的特征串作为 BF 中数据对象进行特征匹配查询[7],以判断攻击事件、追索攻击者.

2.1 系统物理结构

基于 BF 引擎网络取证系统的物理结构设计如图 1 所示. 系统主要包括取证服务器和取证数据处理器两部分. 数据处理器可被嵌入到网络设备(如路由器、网关、交换机)中,主要负责对自己的取证域相当长一段时间内的网络事件(数据包)进行采集、压缩处理和记录,定期将压缩处理后的数据上传到所属取证域的取证服务器. BF 引擎位于数据处理器中. 取证服务器的功能是对各数据处理器收集的数据进行归档,并且负责查询请求的处理和调度,并反馈查询结果.

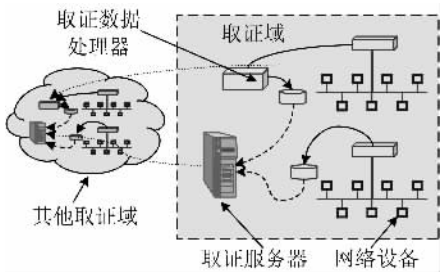

图 1　基于 BF 引擎分布式网络取证系统的物理结构

Fig. 1　Framework of a distributed network forensics system based on BF engine

安装了取证数据处理器的网络设备与取证服务器相连,形成一个层次结构. 在一个取证域中,多个取证数据处理器组成一个网络,有利于共享数据、共享存储和相互协作,它们之间可以采用 P2P 的结构,而不需要中心控制单元. 不同取证域的取证服务器之间也组成网络,进行域间协作,传到某个取证域中的查询被定位到该取证域中的取证服务器上进行处理.

2.2 系统结构与功能描述

基于 BF 引擎分布式网络取证体系结构如图 2 所示. 网络数据包先由数据过滤模块进行过滤,提取

有用的数据包,过滤策略由配置管理模块进行管理控制. 过滤后的数据包交由 BF 引擎,通过 BF 映射压缩处理. BF 的参数配置可由系统管理员通过配置管理模块进行手动配置,也可以由 BF 引擎内部自动管理配置. BF 的参数设置将决定整个取证服务的质量,包括查询的准确率、数据映射的压缩率、BF 的计算复杂度和处理速率等. 映射压缩后的数据由数据处理器上的内存进行管理,先由本地数据处理器存储设备暂时存储,再定期发送到取证服务器上进行归档. 安全管理模块会给每一个写到数据库的数据进行签名和打上时间戳,确保存档数据的完整性. 同时安全管理模块还负责对查询请求进行认证,屏蔽非法欺诈性查询,保证数据安全.

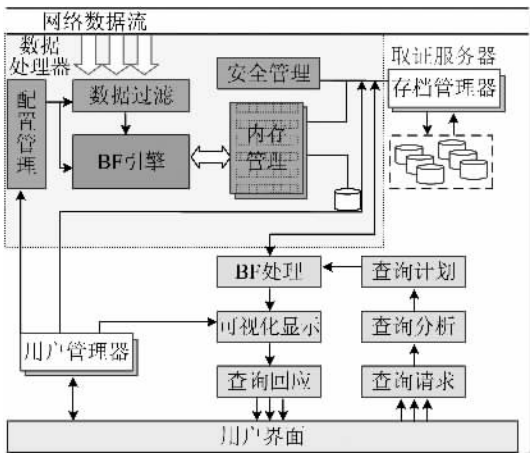

图 2 基于 BF 引擎分布式网络取证系统结构

Fig. 2 Architecture of a distributed network forensics system based on BF engine

用户进行取证查询时,由查询分析模块进行分析,得出查询的范围、查询方法、查询的内容选段等,查询方案由查询计划模块完成. 查询计划生成之后将会定位到某个取证域的取证服务器提交查询请求,读取 BF 相关档案,进行比对查询,以确定这个查询选段(特征选段)是否在网络上进行了传播以及相关联的主机地址,连接记录等信息. 查询结果通过可视化显示反馈给查询用户.

基于 BF 引擎的分布式网络取证系统主要有两个功能:一个是支持针对内容的取证查询,另一个是支持网络攻击事件的取证查询.

针对内容的取证查询需求主要来自军队、政府机关等部门的失泄密事件、非法言论传播事件等,查询的主要原理是通过数据过滤模块提取文字内容相关数据,通过分词规则对文字内容过滤出能够标识内容的关键词(短句),作为 BF 的数据元素,送入 BF 引擎进行数据映射压缩,提交取证服务器存档. 取证查询时由查询分析模块根据查询的文字片断分离出关键词(短句),根据查询的候选主机信息和时间信息,定位查询数据的范围,生成查询计划,进行数据的对比,得出事件是否在网上发生过,相关联的主机信息,发生的时间,所归属的文件信息等,重构、追索事件的发起者.

针对网络攻击事件的取证查询需求主要来自用户对网络攻击事件的追索取证. 取证查询的主要原理是对原始网络数据进行定长切片,由 BF 引擎映射压缩后交由取证服务器存档. 取证查询时根据攻击事件的性质提取攻击特征,例如 MyDoom 蠕虫病毒恶意攻击事件,可以提取病毒邮件附件作为攻击特征. 然后把攻击特征作为查询选段,根据候选主机和时间信息生成查询计划,与映射后的数据进行比较. 如果匹配,则认为病毒在网上传播了一次,根据相关目的源 IP 信息、传输协议信息等分析出发起攻击的地址以及攻击事件的危害程度,重构攻击事件.

2.3 BF 引擎结构设计及关键技术

BF 引擎是整个分布式取证系统的核心部件,其结构原理如图 3 所示. BF 引擎主要负责 BF 的生成和管理、参数的设置、数据映射处理、处理结果的格式化等. 其设计要充分考虑到 BF 中各参数对取证效率和取证结果的影响,在不影响证据有效性的前提下,减少计算复杂度,配置管理安全有效,数据结果支持高效的查询.

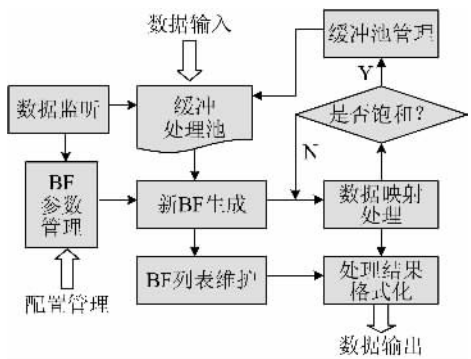

图 3 BF 引擎内部结构原理图

Fig. 3 Schematic of BF engine's structure

关键技术如下:

(Ⅰ) 缓冲处理池. 基于 BF 引擎分布式取证系统是实时取证系统,要对所有的网络数据包进行实时处理,以减少原始网络数据包的存储量. 由于网络数据流量实时变化很大,为了方便数据的映射处理

和参数管理,在 BF 引擎中设计了一个缓冲池,使得数据量在单位时间内趋于稳定. 缓冲池由缓冲管理单元进行管理,以监测缓冲池中的数据元素个数,防止缓冲池中的数据溢出,并配合数据监听模块和数据处理模块,控制缓冲池的大小、数据元素的进出.

(Ⅱ) BF 参数管理. BF 参数主要有 FP,k,m,n,这些参数分别对应取证系统的证据查询准确率、证据数据的压缩率、BF 引擎和查询模块的计算复杂度、BF 引擎的数据吞吐量等. 因此,参数管理将决定整个取证系统的取证质量.

参数管理是在数据监听模块和配置管理模块的共同支持下完成的. 数据监听模块的功能是对缓冲池中的数据量进行监听,得出单位时间内要向 BF 倾倒的数据量,即 n 的数值. 配置管理器是系统管理员维护的参数设置,用来设置压缩率 d 和失误概率 FP,以控制 BF 引擎数据输出的大小和查找的准确率. 在配置管理器中,系统管理员缺省手动配置参数,将由系统自动完成配置管理,参数设置依据是系统软件中预设流量相关参数设置,这是一张预先维护好的与流量相关的参数设置表.

(Ⅲ) BF 生命期维护. 在 BF 引擎当中,需要对一个 BF 的生命期进行维护,也就是要维护其生成、添加数据、生命周期结束的存储纪录. 在缓冲池里,把一段时间内的数据元素添加到 BF 中,因此生成 BF 也是以时间周期为标记的. 例如,以 60 s 作为时间周期,把 60 s 内的数据倾倒到一个 BF 中,当 BF 的数据饱和了,就需要新生成一个 BF,或者 60 s 内的数据倾倒完成了,就需要倾倒下一个 60 s 的数据对象. 倾倒完成之后,还需要在 BF 列表维护部件中进行注册,列表维护部件是一个逻辑部件,所完成的功能是对 BF 进行时间区分标记、参数设置、Hash 函数进行注册,以方便日后查找.

3　系统性能分析和取证过程描述

为了分析系统的压缩性能和取证准确率,我们在局域网上模拟了整个取证数据采集映射存储、取证查询的过程. 我们把取证服务器和数据处理器都运行在一个配置为 P4-3.2 GHZ,1 G 内存、硬盘 200 G 的台式 Dell 计算机上,局域网内机器数量为 10 台,局域网与互联网连接. 其中一台主机作为未知病毒攻击发起者(病毒以邮件形式传播,病毒副本在大小为 22 kB 的附件里),三小时内共向局域网五台机器以及局域网外部机器发送病毒样本 15 次. 取证的目的是追踪传播病毒路径,通过取证查询定位发送传播的用户,并确定局域网内哪些机器被感染,以便采取应急措施.

数据采集时,把网卡设为混杂模式,对网络层数据抓包采集. 假肯定率 FP≤1%,块大小 $b=128$ 字节. 由参数管理模块得出 $k=7$(取整),$m/n=9.59$. 根据数据流量的统计,平均每分钟向 BF 引擎注入 10 000个大小为 128 字节的数据元素,$m=8\ 000$字节,则单个 BF 的 n 值应该小于 800 才能保证假肯定率的大小. 每隔 5 s 生成一个 BF 或者数据元素的个数超过 800(5 s 内)重新生成新的 Bloom filter. 共对三小时的网络数据进行记录,原始网络数据大小约为 150 MB,映射压缩性能理论实际值比较如表 1 所示.

表 1　压缩性能比较

Tab. 1　Comparison of compressibility

比较对象	大小	BF 个数	压缩率
理论压缩后	17.28 MB	2 160 个	88.5%
实际压缩后	32.56 MB	3 112 个	78.3%

通过表 1 中的压缩率比较,我们可以看出,公式计算的理论值压缩率高达 88.5%,但是实际压缩率只有 78.3%. 这是由于理论上每个 BF 的数据元素的个数都应该是 800 个左右,这样,原始数据的大小应该是 221 MB,而不是实际的 150 MB. 其次网络流量并不均匀,某时间段内网络数据流量偏大,会生成多个 BF,为的是保证 FP 小于 1%. 最后实际映射后的单个 BF 大小要大于8 000字节,还要存储注册 BF 信息、时间标识、候选目的源 IP 地址以及传输协议等必要信息,造成了实际压缩率小于理论值.

取证查询分析时,缺省了候选主机的选择和时间选择,在大规模应用时,选择查询时间段和候选主机会提高查询效率,缩小查询的范围. 攻击特征选取病毒副本的附件部分(共 22 kB)的 0 字节到 256 字节作为查询选段,查询的时候以 128 字节为单位,从 0 字节依次向右位移切片,共需要位移 129 次,产生了 129 个候选查询选段. 每个查询选段进行 BF 映射,与3 112个 BF 数据进行比较,查询结果如表 2 所示.

从数据来看,发起攻击次数的实际值反而比查询结果还要多,并没有出现理论上的假肯定现象. 主要原因有两种可能:一是每 5 s 生成一个 BF,在此期间如果同时向两个地址发送邮件,两个病毒样本就会在同一个BF里,系统默认为一次;另一种可能

表 2 取证查询准确率

Tab. 2 Validity of forensics system

比较对象	发起攻击次数	接收病毒次数	查询效率
实际值	15	5	在没有优化查询计划的情况下,对于单个 BF 的元素映射后仅需比较 129 次,原始数据比较需要上百万次
查询结果	13	5	

就是在一次攻击事件中,病毒样本被分别存放在不同的两个 BF 中,导致了查询的误差. 文献[8]提出了带偏移量的基于块的 BBF(block-based BF)支持取证中对选段查询分析和 HBF(hierarchical Bloom filter),解决了多包查询的问题.

通过对查询到的 13 个 BF 存储的目的源 IP 地址分析,共产生了 5 个不同的 IP 地址,其中都出现的 IP 地址只有一个,为攻击发起者,达到了追踪攻击源的目的. 接收病毒次数和实际值相同.

从实验结果来看,BF 数据结构能够很好地对网络原始数据进行映射压缩存储,压缩比率达到 78% 以上,查询效率也大大提高,查询的准确率也较高,存储后的数据也基本上能够反映网络事件的真实情况,解决了海量数据传输和存储带来的资源瓶颈问题.

4 结论

本文研究和设计了一种基于 BF 引擎的分布式网络取证系统,详细介绍了 BF 引擎的结构设计和关键技术. 该系统对网络原始数据用 BF 引擎进行压缩存储,包含了足够的事后分析查询信息,却大大减少了原始数据的存储量,支持高效的事后取证查询分析,而且系统采用了模块化设计,将来可以作为网络安全器件的一个标准配置广泛应用于网络.

参考文献(References)

[1] 张有东,王建东,叶飞跃,等. 网络取证及其应用技术研究[J]. 小型计算机系统,2006, 27(3): 558-562.

[2] 杨泽明. 计算机与网络取证系统的研究与实现[D]. 中科院高能物理研究所,2007.

[3] 徐小琴,龚俭,周鹏. 网络取证系统及工具分析[J]. 微机发展, 2005, 15(5): 139-141.

[4] Bloom B. Space/time trade-off in hash coding with allowable errors[J]. Communication of the ACM, 1970, 13(7): 422-426.

[5] Bloom filter-the math[EB/OL]. http://www.cs.wisc.edu/-cao/papers/summary-cache/node8.html.

[6] 肖明忠,代亚非. Bloom filter 及其应用综述[J]. 计算机科学,2004, 31(4): 180-183.

[7] Dharmapurikar S, Attig M, Lockwood J. Design and implementation of a string matching system for network intrusion detection using FPGA-based bloom filters[R]. Department of Computer Science and Engineering, Washington University, 2004.

[8] Shanmugasundaram K, Memon N, Savant A, et al. Fornet: a distributed forensics network[EB/OL]. http://isis.poly.edu/kulesh/skunk/pubs/mmm-acns-2003.pdf.