软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(10):3295−3308 [doi: 10.13328/j.cnki.jos.005800]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

多节点系统异常日志流量模式检测方法*

王晓东[1,2], 赵一宁[1], 肖海力[1], 迟学斌[1,2], 王小宁[1]

[1](中国科学院 计算机网络信息中心,北京 100190)

[2](中国科学院大学,北京 100049)

通讯作者: 赵一宁, E-mail: zhaoyn@sccas.cn

摘 要: 随着国家高性能计算环境各个节点产生日志数量的不断增加,采用传统的人工方式进行异常日志分析已不能满足日常的分析需求.提出一种异常日志流量模式的定义方法:同一节点相同时间片内日志类型的有序排列代表了一种日志流量模式,并以该方法为出发点,实现了一个异常日志流量模式检测方法,用来自动挖掘异常日志流量模式.该方法从系统日志入手,根据日志内容的文本相似度进行自动分类.然后将相同时间片内日志各个类型出现的次数作为输入特征,基于主成分分析的异常检测方法对该输入进行异常检测,得到大量异常的日志类型序列.之后,使用基于最长公共子序列的距离度量对这些序列进行层次聚类,并将聚类结果进行自适应 K 项集算法,以得出不同异常日志流量模式的序列代表.将国家高性能计算环境半年产生的日志根据不同时间段(早、晚、夜)使用上述方法进行分析,得出了不同时间段的异常日志流量模式和相互关系.该方法也可以推广到其他分布式系统的系统日志中.

关键词: 异常日志流量;主成分分析;层次聚类;最长公共子序列;自适应 K 项集算法

中图法分类号: TP316

中文引用格式: 王晓东,赵一宁,肖海力,迟学斌,王小宁.多节点系统异常日志流量模式检测方法.软件学报,2020,31(10):3295−3308. http://www.jos.org.cn/1000-9825/5800.htm

英文引用格式: Wang XD, Zhao YN, Xiao HL, Chi XB, Wang XN. Multi-node system abnormal log flow mode detection method. Ruan Jian Xue Bao/Journal of Software, 2020,31(10):3295−3308 (in Chinese). http://www.jos.org.cn/1000-9825/5800.htm

Multi-node System Abnormal Log Flow Mode Detection Method

WANG Xiao-Dong[1,2], ZHAO Yi-Ning[1], XIAO Hai-Li[1], CHI Xue-Bin[1,2], WANG Xiao-Ning[1]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: With the increasing number of logs produced by nodes in CNGrid, traditional manual methods for abnormal log analysis can no longer meet the need of daily analysis. This study proposed a method to define the abnormal log traffic pattern: The orderly arrangement of log types in the same node and at the same time slice represents a log traffic pattern. Based on this method, a log traffic pattern detection method was implemented, which was applied in automatically mine of abnormal log traffic pattern. The method starts with system log and classifies automatically according to the text similarity of log content. Then, the frequency of each types of log in the same time slice is taken as the input feature, and the anomaly detection method based on principal component analysis (PCA) is used to detect the abnormal input, and a large number of abnormal log type sequences are obtained. A distance metric based on the longest common subsequence is used to cluster these sequences by hierarchical clustering method. The clustering results are used with the adaptive *K*-itemset algorithm to get the deputies of the abnormal log flow modes. The above method was used to analyze the logs generated in the national high performance computing environment CNGrid in half a year according to different time periods (morning,

* 基金项目: 国家重点研发计划(2018YFB0204002); 国家自然科学基金(61702477)

Foundation item: National Key Research and Development Program of China (2018YFB0204002); National Natural Science Foundation of China (61702477)

收稿时间: 2018-06-08; 修改时间: 2018-09-10; 采用时间: 2018-12-27; jos 在线出版时间: 2019-11-06

CNKI 网络优先出版: 2019-11-06 11:48:54, http://kns.cnki.net/kcms/detail/11.2560.TP.20191106.1148.002.html

night, midnight), and has obtained the abnormal log traffic patterns and their relationships in different time periods. The method can also be extended to the system logs of other distributed systems.

Key words: abnormal log flow; principal component analysis; hierarchical clustering; longest common subsequence; adaptive *K*-itemset algorithm

系统日志是由Linux系统syslog服务产生的日志,用于记录系统中发生的各类事件信息,在大规模系统中具有很重要的意义.中国国家高性能计算环境是由国内众多超算中心和高校的计算集群组成的国家级大型高性能计算环境,采用中国科学院计算机网络信息中心自主研发的网格环境中间件 SCE[1]聚合了大量的通用计算资源,为全国众多高校和研究机构的用户提供了优质的计算服务.由于环境的系统中存在种类繁多的应用和服务,使得每个节点都会产生大量的系统日志,这使得最终的日志文件变得极为庞大,直接采用人工的方法处理这些日志显然是一项不可能的任务.而这些日志中往往包括各种异常现象的信息,为了从大量的日志中提取出这些异常的流量信息,我们需要使用一些异常检测方法对其进行分析.

传统的流量异常检测方法基于签名,这种方法有一个显著的局限性,它们的本质是利用以前出现过的攻击来过滤发现异常,即它们无法检测出可能会出现的新异常.此外,即使一个新的攻击被检测到了,对应的签名也得到了开发,但是该签名在网络上的部署通常会有延迟.这些限制使得基于机器学习的入侵检测技术越来越受到关注.基于机器学习的检测技术主要分为两大类:误用检测和异常检测[2].误用检测属于机器学习中的有监督算法,这种算法在使用时需要将所有训练样本标记为“正常”和“异常”,然后通过输入这些有标签的数据进行学习训练,更新模型参数.当新的流量特征输入时,就可以通过学习到的模型预测对应时刻的流量为“正常”还是“异常”.误用检测的主要优点是在检测已知类型的异常时具有很高的精度,但其缺点是无法检测未知的流量异常.而异常检测在使用时不需要提前得到数据的标签类型,属于无监督学习算法.其主要思想是:根据已知的大量数据进行模型建立,然后自动地从这些数据中找到偏离数据中心的少量数据作为潜在可能的异常.因此,此类异常检测技术可以自动地识别任何新的异常,但其缺点是可能出现误判的情况,即将正常的流量标记为异常.高性能计算环境中,多节点产生的日志复杂,内容丰富,迭代性强,因此使用无监督学习算法较为合适.

Vaarandi[3]提出了一个轻量级的、开源的、与平台无关的基于规则的事件相关的工具,称为 SEC(简单事件相关器),并在文献[4,5]中描述了它的应用经验.该工具的主要思想是:通过写配置文件对日志中的不同日志类型进行匹配,然后通过匹配的类型出现顺序来进行模式判定.比如匹配出一条端口扫描的日志,随后匹配出一条防火墙生成的拒绝日志,则代表有人企图访问一台主机的一种流量异常.因此,不同类型的日志按照一定顺序出现,可能代表一种异常的流量模式.其不足之处是:所有配置文件都必须自己完成,对于不同的日志系统都需要修改配置文件.

本文在对日志进行分析时,首先自动地将日志分类,然后通过无监督异常检测方法得到异常日志类型的序列,并将这些序列所代表的异常日志流量模式进行聚类,最后分析结果并得出结论.本文的整体流程除了设定少量阈值外,得到异常日志流量模式的过程全自动化进行,这样使得系统管理员对系统日志流量的监控变得简单.

本文第 1 节介绍相关研究内容.第 2 节描述系统日志前期的分类研究和后续讨论中使用的数学符号.第 3 节对整体方法进行描述.第 4 节设置实验并对实验结果进行分析.最后在第 5 节进行总结和展望.

1 相关研究

本节简单讨论一下日志模式分类、日志后验分析以及日志类型序列度量的相关研究.

(1) 日志模式分类

Vaarandi[6]对于日志文件数据展示了一种创新的聚类算法 SLCT,该聚类算法基于 Apriori 频繁项集,需要使用者提供支持阈值作为输入.这个支持阈值既要控制这些算法的输出,同时也是其内部实现机理的基础.之后,他又在文献[7]中改进了日志的聚类算法,得到名为 LogHound 的算法.这一算法是一种基于广度优先搜索的频繁项集挖掘算法,用于从事件日志中挖掘频繁模式.该算法结合了广度优先和深度优先算法的特点,同时考虑了

事件日志数据的特殊属性,因此比 SLCT 更接近地反映 Apriori 算法.SLCT 和 Loghound 两种算法都将那些不匹配任何频繁模式的日志分类为离群值.Makanju 等人[8]引入了 IPLoM 算法,这是一种用于挖掘事件日志簇的新算法.与 SLCT 不同的是,IPLoM 是一种层次聚类算法,它以整个事件日志作为单个分区开始,并在 3 个步骤中迭代分割分区.与 SLCT 类似,IPLoM 将事件日志行中的位置视为单词,因此对单词位置的移位敏感.由于其分层性,IPLoM 不需要支持阈值,而是需要其他一些参数(如分区支持阈值和簇优度阈值),这些参数对分区的划分进行了细粒度的控制.IPLoM 相对于 SLCT 的一个优点是能够使用通配符尾部(例如 Interface * *)来检测日志行模式,其作者将 IPLoM 算法与 SLCT 和 LogHound 算法进行比较,并得出其效果优于上述两种聚类算法的结论.此后,Vaarandi 和 Pihelgas[9]提出了 logcluster 算法,该算法对于文本事件日志进行数据聚类和日志行模式挖掘,并修复现有聚类算法的一些缺点.而本文使用基于字符匹配的分类算法,针对于国家高性能计算环境的系统日志进行分类,代码压缩率和后续特征创建都显示出了不错的效果.

(2) 日志后验分析

一些学者在分析日志寻找相关异常方面进行了研究,比如,Xu 等人[10]通过源代码匹配日志的格式找出相关变量,通过词袋模型提取对应日志变量的特征,然后使用这些特征,通过主成分分析方法降维,根据主成分分析的最大可分性检测异常的日志文件,最后使用决策树可视化该结果.Fronza 等人[11]使用随机索引为代表的操作序列,根据其上下文为每个日志中的操作特征化,然后使用支持向量机关联序列到故障或无故障的类别上,以此来预测系统故障.Weiss 等人[12]研究了从有标签特征的事件序列中预测稀少事件的问题,基于遗传算法的机器学习系统,能够在预测稀有任务上达到比较好的结果.Yamanishi 和 Maruyama[13]提出了一种新的动态系统日志挖掘方法,以更高的置信度检测故障症状,并发现计算机设备间的连续报警模式.Yuan、Mai 和 Xiong[14]提出了一个名为 Sher-Log 的工具,它利用运行时日志提供的信息来分析源代码,以推断在失败的生产运行期间必须或可能发生的事情.它不需要重新执行程序,也不需要知道日志的语义.它推断关于执行失败的控制和数据值信息.Peng、Li 和 Ma[15]应用文本挖掘技术将日志文件中的消息分类为常见情况,通过考虑日志消息的时间特性来提高分类的准确性,并利用可视化工具来评估和验证用于系统管理的有趣的时间模式.Wang[16]也使用机器学习方法对日志进行后验分析,但是主要突出的是日志线上的可视化展示.本文使用文献[10]中描述的异常检测方法,但是输入的日志类别对应于基于字符匹配生成的分类结果,同时,研究对象为日志类型的有序排列,在得到未知的异常日志流量模式上更有优势.

(3) 日志类型序列度量的研究

日志类型序列的分类问题,实际上可以归类为字符匹配模式.在字符匹配中,为了判别两个序列的相似度,Hamming[17]在对两个字符串之间进行比较时使用了最直接的一对一匹配法,以两个序列匹配的字符数目作为两个序列之间的相似度衡量指标.Damerau 和 Levenshtein[18]使用两个序列之间的最小操作数(包括序列的插入、删除、替换或两个相邻字符之间的转换)作为两个序列之间的相似度衡量指标.在序列对比方面,Needleman[19]和 Smith[20]分别对序列进行全局对比和局部对比,用以得到最符合两个序列的全局序列和局部序列.本文在选择日志序列特征时考虑到了字符串序列的特点.根据 Seker[21]在字符串匹配算法中对人名字符度量上选择的方法,即前 K 个频繁出现次数的字母项和其对应数量作为识别序列的关键要素.将不同类型的日志抽象为不同的字符,根据其特点实现自适应 K 值的挑选,在日志类型序列的比较上更为适合.

2 背 景

2.1 系统环境日志前期分类研究

日志预处理需要完成的主要任务是如何将日志内容压缩并分类,这样做既可以减少数据量,又有利于后续使用相关机器学习算法进行分析.我们在前期工作中,使用了一种基于字符匹配的方法对系统日志的分类问题进行研究,主要方法是根据字符匹配原则对系统日志进行分类[22].日志模式匹配的核心思想是字符一一对应,即:将整体匹配的字符数与总字符数进行比值,并将得到的数值与设定的阈值进行对比,如果结果超过设定的阈值,则将这两行日志定义为一类[22].

我们在对日志类型进行字符对比分类时,为了减少分类的模式,在字符匹配时引入了最长公共子序列的概念[23],这样使得计算时得到的匹配模式大量减少.得到分类结果后,可以验证日志的压缩率达到 99%以上.

2.2 后续内容中基本符号解释

后续系统设计和分析时,我们为了简化讨论,将定义一些数学符号,具体解释见表 1.

Table 1 Mathematical notation and interpretation

表 1 数学符号与解释

符号	解释
T_n	第 n 种类型的日志
t_n	第 n 个时间片段
λ	矩阵的特征值
V	矩阵的特征向量
NM_n	节点的第 n 种异常日志流量模式
S_n	得到的第 n 个日志类型有序排列
D_n	白天时间段(8:00~18:00)日志产生的第 n 种类型序列
N_n	晚上时间段(18:00~24:00)日志产生的第 n 种类型序列
M_n	深夜时间段(0:00~8:00)日志产生的第 n 种类型序列

3 方法结构

为了适应高性能计算环境下出现的流量异常检测问题,我们根据系统中产生的日志特性,选择适当的预处理和无监督机器学习算法来对系统日志的分析过程进行整体的构建与规划,设计出一种异常流量模式检测的方法,该方法的结构主要包含 3 部分:预处理模块、异常处理模块、分类模块.各部分的功能简述如下.

(1) 预处理模块:根据日志匹配算法对输入的大量日志进行分类,输出日志类别文件;

(2) 异常处理模块:根据输入的待分析日志以及上一步得到的日志类别文件,使用基于主成分分析的异常检测方法,得出异常的时间片内不同节点的异常类型序列;

(3) 分类模块:根据输入的大量异常类别序列,使用基于最长公共子序列的距离度量进行层次聚类,得出不同的异常日志流量模式.

根据各个功能的作用,我们得到日志整体结构图,如图 1 所示.

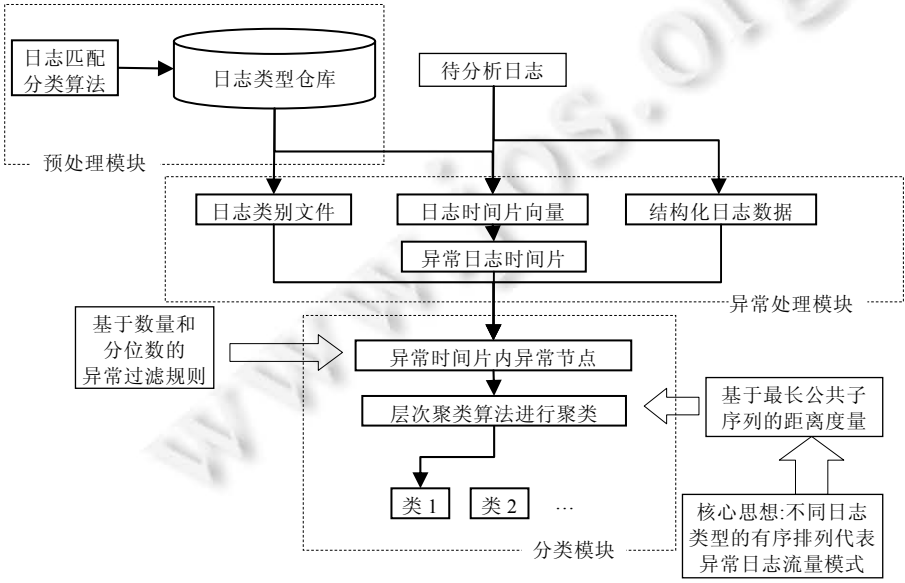


Fig.1 Flow chart of log overall structure

图 1 日志整体结构流程图

3.1 预处理模块

异常流量模式检测方法进行预处理的目的是将日志类型记录在本地的日志类型仓库中,以便后期分析.日志类型提取的具体方法[23]是读取一行日志,并与日志仓库中的日志类型逐一进行比较.设该条日志 l 包含 n 个单词,日志仓库中取出的待比较日志类型 l' 包含 m 个单词,则定义 l 与 l' 的相似度为

$$S(l,l')=\frac{|LCS(l,l')|\times 2}{n+m},$$

其中,$|LCS(l,l')|$ 表示 l 与 l' 的最长公共子序列的单词数量.计算出 $S(l,l')$ 后,即可将该值和预定义阈值 $t(0<t<1)$ 进行比较:如果结果小于该阈值,则将该类型加入到日志类型仓库中;否则,读取下一条日志进行比较.其处理流程如图 2 所示.

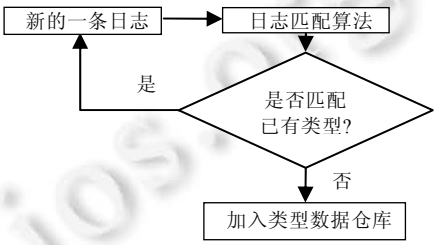


Fig.2 Flow chart of preprocessing module

图 2 预处理模块流程图

需要指出的是:预处理模块分析的对象是所有待分析的系统日志,并根据这些日志进行模式匹配,得到不同的日志类型,然后根据这些日志类型建立日志仓库.其目的是化简日志类型的维度,本身并没有过滤掉任何待分析的日志.在后续进行实际流量分析时,可以将每个时间片内出现的日志与日志仓库中的日志类型进行对比,抽象出不同类型的日志在对应时间片内出现次数的一个向量,以利于使用相关机器学习算法进行分析.

3.2 异常处理模块

日志的异常处理输入需要用到上一小节得到的日志类型.我们首先根据日志仓库中不同日志的类型生成日志类别文件,然后基于该日志类别文件,将需要具体分析的日志文件进行特征创建,得到日志时间片向量,之后根据待分析日志的格式特点将日志文件的每一行数据按照数据表的格式存储,得到结构化的日志数据,最后将上述得到的 3 个文件加载,并使用无监督异常检测方法进行分析,即可得到对应具有异常流量特征的时间片.具体实现方法如下.

(1) 特征创建

我们首先在内存中加载待分析日志,然后根据日志的时间计算出时间戳,在得到时间戳后,即可根据时间戳的大小将日志按时间顺序进行排序,并根据实际情况把日志按照给定的时间片进行拆分,每一个时间片可以作为一个样本的输入向量.该向量的每一个维度代表一种类型的日志,而其对应的数值等于该类型在当前时间片段内出现的次数.例如:如果取时间片为 5 分钟,并且假设在 t_1 时间片内得到的日志可见表 2.

Table 2 Log in t_1 time slice

表 2 t_1 时间片内得到的日志

日期时间	时间戳	节点名	日志内容	所属类型
Jun 25 08:01:04	15120064000	*Node*1	Connection closed by ⟨IP⟩	T_1
Jun 25 08:01:08	15120068000	*Node*2	Accepted publickey for usr1 from ⟨IP⟩	T_0
Jun 25 08:01:08	15120068000	*Node*1	Received disconnect from ⟨IP⟩	T_3
Jun 25 08:01:20	15120080000	*Node*3	Received disconnect from ⟨IP⟩	T_3
Jun 25 08:02:01	15120121000	*Node*2	session opened for user usr1 by (*uid*=0)	T_2
Jun 25 08:02:28	15120148000	*Node*1	Connection closed by ⟨IP⟩	T_1
Jun 25 08:02:30	15120150000	*Node*2	Connection closed by ⟨IP⟩	T_1
Jun 25 08:02:31	15120151000	*Node*3	Connection closed by ⟨IP⟩	T_1

根据该时间片内不同类型日志出现的次数,可以得到的样本输入向量见表 3 中的 t_1 行.

Table 3 Log type's vector of time slice

表 3 时间片日志类型向量

时间片	T_0	T_1	T_2	T_3	…	T_n
t_1	1	4	1	2	…	0
…	XX	XX	XX	XX	XX	XX
t_m	XX	XX	XX	XX	XX	XX

表 3 代表待处理日志经分析后得到的输入矩阵,其中,n 代表预处理模块中使用日志模式匹配得到的日志类型数量,m 代表待分析日志按照固定时间片分割的时间片个数.假设待分析日志的总时间跨度为 T,则可得到 $m=\lfloor T/t \rfloor$.根据上述方法即可将待分析日志转换为 m 行、n 列的数据矩阵,该矩阵每行代表一个样本数据,每列代表一个样本的特征段位.

(2) 基于主成分分析的流量异常检测

主成分分析是一种常用的机器学习算法,该算法在推导过程中使用了最近重构性原理,即:将高维的数据映射到低维空间中,使得每个高维空间中的数据映射到低维空间后的距离之和达到最小.这样做使得原始数据投影到低维空间时的距离最小,所以当找到这个最近重构的低维空间后,计算出原空间样本点到低维空间的距离,该距离即可以作为异常与否的度量标准.即:该距离越大,原来数据更可能是异常数据.该方法的具体实现步骤如下.

步骤 1 将 m 行 n 列的数据矩阵 A 输入,将所有数据中心化得到矩阵 B.

步骤 2 解得 B^TB 的特征值 $\lambda_i(i=0,1,\ldots,n)$.注意:这里需要自己选择降维后的方差比重,这里我们选择 90% 为方差比重.带入计算使得 $\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{j=1}^{n}\lambda_j}>90\%$.

步骤 3 将得到数值最高的 k 个特征向量 $V_1,V_2,\ldots,V_k$ 组成矩阵 $P=[V_1,V_2,\ldots,V_k]$,计算该矩阵的正交投影矩阵为 $V_P=P(P^TP)^{-1}P^T=PP^T$.

步骤 4 如果原来的向量为 y,则该向量到其映射的子空间的欧几里德距离可以通过计算平方预测误差 $SPE=\|y_a\|^2$ 得到.注意:其中,y_a 是 y 到异常子空间 s_a 上的投影,并且可以通过式子 $y_a=(I-V_P)y=(I-PP^T)y$ 计算得到.

步骤 5 根据上述算法计算出每一个点到子空间的距离,将其与检测阈值 Q_α 进行比较:如果 $SPE=\|y_a\|>Q_\alpha$ 则标记 y 是异常的.其中,Q_α 表示在 $(1-\alpha)$ 置信水平下 SPE 残差函数的阈值统计量.基于主成分分析检测的方法来源于文献[10],其检测的异常数值 Q_α 可用如下公式得到[24]:

$$Q_\alpha^2=\phi_1\left[\frac{C_\alpha\sqrt{2\phi_2h_0^2}}{\phi_1}+1+\frac{\phi_2h_0(h_0-1)}{\phi_1^2}\right]^{\frac{1}{h_0}},\text{其中},\phi_i=\sum_{j=r+1}^{m}\lambda_j^i(i=1,2,3),h_0=1-\frac{2\phi_1\phi_3}{3\phi_2^2}.$$

公式中的 λ_j 代表样本数据协方差矩阵第 j 个主成分投影在子空间的特征值.C_α 表示标准正态分布的 $1-\alpha$ 百分位数.

上述方法进行异常检测的基本原理是:系统在产生不同类型的日志时,在正常情况下,各个类型日志出现的比例趋于稳定.如果各个类型日志在某些时间段内出现大量的比例失衡的情况,则很有可能在这个时间片内有异常的情况发生,则这些时间片段需要重点关注和分析.

3.3 分类模块

日志异常模式是由不同类型的日志有序连接而成的.单个节点的异常日志流量模式可定义为日志类型的有序排列.单独一种日志类型的出现并不能说明什么问题,比如单独出现一次 T_7(认证失败)类型的日志,可能是由于用户不小心密码输入错误引起的.但是如果在一定的时间片内,同一个主机频繁出现 T_7(认证失败)类型日志、T_{11}(密码错误)类型日志、T_3(连接断开)类型日志,则说明该类型序列可能是一种暴力破解登录的尝试.因此,

异常模式提取的思想就是在异常时间片内找到同一个主机产生的日志类型的关联规则,然后得到异常日志流量模式.

(1) 异常流量筛选

将日志时间片向量带入上一小节中介绍的方法,即可得到日志流量中对应的异常时间片.我们统计出正常、异常时间片中不同类型日志出现数量的分布,计算出其对应正常、异常日志和出现数量的中位数来进行后续比较.根据比较的差值来得出正常、异常时间片主要差异的日志类型.

为了将时间片内所有的日志类型序列得到,我们单独将每个异常时间片进行抽离.对于每个单独的异常时间片,根据不同节点将对应的日志类型序列记录下来.比如某个时间片内出现了异常,我们将时间片内的全部日志提取出来,根据这些日志数据得到所有节点名,之后根据节点名单独挑选出每个节点在对应时间片内的所有日志类型,组成一个日志类型序列.这些序列中有许多并非异常流量的数据,因此我们根据两个规则进行异常类型序列的筛选,具体规则如下.

A 该主机日志数量超过给定阈值 HTN(host threshold number);

B 该主机日志的类型包含前面检测出的主要差异的日志类型.

选择机制 A 的主要原因是:如果时间片内出现的日志数量比较少,则说明该节点并没有发生流量异常现象,因此可以不予考虑.选择机制 B 的主要原因是,根据正常异常类型分布得到的是最能区别出正常与异常的日志类型.

例如:检测出的某一个异常时间片内出现的日志见表 2,则根据节点名得到 3 个不同的日志类型序列:*Node*1 产生的序列 $S_1=T_1,T_3,T_1$,*Node*2 产生的序列 $S_2=T_0,T_2,T_1$,*Node*3 产生的序列 $S_3=T_3,T_1$.假如设定的阈值为 $HTN=2$,根据中位数得到的异常比正常多的日志类型为 T_3,则根据过滤规则 A 将会把 S_3 过滤掉,根据过滤规则 B 将会把 S_2 过滤掉,这样过滤后留下的日志类型序列是 S_1.

使用这两种过滤机制筛选后,依然可以得到大量的异常流量类型序列.其中,各个序列之间有大量重复的流量序列特征,因此需要对这些日志类型序列进行模式分类,建立异常日志流量模式库.

(2) 基于最长公共子序列距离度量的层次聚类方法进行异常流量分类

层次聚类法是一种基于距离度量的聚类算法,其特点是可以根据距离阈值的变化得到原始数据的分类数目,因其显示效果和解释效果好,而在机器学习中非常常用.我们使用基于最长公共子序列的相似度算法对两个日志类型序列计算距离,然后将上一小节得到的大量异常流量类型序列进行层次聚类.使用最长公共子序列度量两条日志类型序列,是因为日志类型的有序排列可以决定一个节点的异常日志流量模式.在实际运算时,我们认为最长公共子序列匹配的日志类型数目越大,则代表两条日志序列匹配的效果越好.为了使得两个类型序列的相似度越高,距离越近,我们以如下方法定义两条日志类型序列的距离.

设两个待比较日志类型的序列为 S_1 和 S_2,通过最长公共子序列得到匹配的类型序列为 S,则距离计算公式为

$$d=\begin{cases}\dfrac{|S_1|}{|S|}+\dfrac{|S_2|}{|S|}, & |S|>0\\ |S_1|+|S_2|, & |S|=0\end{cases}$$,其中,$|S|$代表类型序列S的长度 .

例如,有两条日志类型序列分别是 $S_1=T_7,T_{11},T_7,T_{11},T_3,T_2,T_1$ 和 $S_2=T_7,T_{11},T_7,T_{11},T_2$,则对这两条日志类型序列进行最长公共子序列比较后的结果见表 4.

Table 4 Comparison results of the longest common subsequence

表 4 最长公共子序列比较结果

	T_7	T_{11}	T_7	T_{11}	T_3	T_2	T_1
T_7	***+1***	+1	+1	+1	+1	+1	+1
T_{11}	+1	***+2***	+2	+2	+2	+2	+2
T_7	+1	+2	***+3***	+3	+3	+3	+3
T_{11}	+1	+2	+3	***+4***	***+4***	+4	+4
T_2	+1	+2	+3	+4	+4	***+5***	***+5***

可以看出,得到的匹配序列为 $S=T_7,T_{11},T_7,T_{11},T_2$.根据上述公式,我们可以得到当前变量:$|S_1|=7,|S_2|=5,|S|=5$,则计算得到这两个序列之间的距离为 $d=\left(\frac{7}{5}+\frac{5}{5}\right)=\frac{12}{5}=2.4$.

3.4 基于自适应K项集的标准进行类别选择

层次聚类中,我们选择的距离度量仅能计算出两条日志类型序列的距离,但无法将任意一个日志类型的序列映射到向量空间,这样就无法通过求日志类型序列的几何中心得到中心位置的日志类型序列.因此,我们需要从其他角度来处理这个问题.

日志类型序列可以看成字符串数据.Seker[21]在使用字符串匹配算法匹配人名时,通过选择人名中前 K 个频繁出现次数的字母项和其对应数量作为识别人名的关键要素.我们也可以使用相似的方法来确定日志类型序列的中心位置和单条日志距离其中心位置的距离.即,将同一种异常日志流量模式中所有出现的日志类型数量的平均值作为该异常日志流量模式的中心数量.在寻找最靠近中心数量的日志类型序列时,考虑到日志序列类型较多,数量分布不均匀的特点,因此不适合使用原文中固定 K 值法对序列进行比较.在实践中,我们提出一种自适应 K 值的算法,该算法在计算不同类型日志中心距离时,采用了平均流量的数值进行比较,而对进行比较的类型数目 K 的选择采用如下公式:

$$K=CountIf(AverageNum_x>AverageTotalNum\times Threshold),x=1,\ldots,n.$$

$AverageNum_x$ 代表第 x 类型日志在该异常日志流量模式中出现次数的平均值,$AverageTotalNum$ 代表所有类别日志在该异常日志流量模式中出现的平均值之和,$Threshold$ 代表异常百分比阈值.在实际操作时,我们将 $Threshold$ 取值为 $\frac{1}{HTN}$.这样选择是因为在异常流量筛选时,我们确定的日志数量阈值为 HTN,即保证了筛选出来的每条日志异常流量所包含的日志数目大于 HTN.因此,设定 $Threshold=\frac{1}{HTN}$ 就保证挑选出的前 K 个频繁类型的日志数量至少超过 1 条.因为如果一种类型的日志数量低于 1 条,则显然为不频繁类型.日志数目超过 1 条证明如下:

$$AverageNum_x > AverageTotalNum\times Threshold > HTN\times\frac{1}{HTN}=1.$$

在确定 K 值后,即可根据平均流量计算每条日志模式序列与中心流量之间的距离,第 m 个日志类型序列与中心之间的距离公式为

$$d_m=\sum_{i=1}^{K}|\,AverageNum_i-TypeNum_{mi}\,|,$$

其中,$AverageNum_i$ 代表第 i 种类型日志在该日志类型序列集合中出现数量的平均值,$TypeNum_{mi}$ 代表第 m 个序列中第 i 种类别日志出现的数量.根据距离公式将所有距离值计算出来后,将所有序列计算的 d 值中结果最小的序列作为该类型异常流量的特征序列代表.

假设上一小节聚类后得到的 m 类异常日志流量模式记为 $\{Y^1,Y^2,\ldots,Y^m\}$,第 m 类异常日志流量模式 Y^m 包含的日志类型序列的集合记为 $\{S_1^{(m)},S_2^{(m)},...,S_{nm}^{(m)}\}$, 则自适应最大 k 项集算法具体步骤如下.

步骤 1　读取第 w 类日志流量模式的全部日志类型序列集合 $Y^w=\{S_1^{(w)},S_2^{(w)},...,S_{nw}^{(w)}\}$;

步骤 2　统计 Y^w 包含的所有日志类型 $\{T_{w1},T_{w2},\ldots,T_{wt}\}$;

步骤 3　计算数量阈值 $AT=\frac{\sum_{i=1}^{nw}|\,S_i^w\,|}{nw}\times Threshold=\frac{\sum_{i=1}^{nw}|\,S_i^w\,|}{nw}\times\frac{1}{HTN}$;

步骤 4　得到前 K 种频繁日志类型 $\{T_1^{(w)},T_2^{(w)},...,T_K^{(w)}\}\in\{T_{w1},T_{w2},...,T_{wt}\}$, 满足 $T_x^{(w)}(x=1,...,K)$ 在集合 Y^w 包含的日志类型序列中出现数量的平均值大于阈值 AT;

步骤 5　根据公式 $d_m=\sum_{i=1}^{K}|\,AverageNum_i-TypeNum_{mi}\,|$ 计算所有序列 $\{S_1^{(m)},S_2^{(m)},...,S_{nm}^{(m)}\}$ 对应的距离 $\{d_1,d_2,\ldots,d_{nw}\}$;

步骤 6　得到最小距离 $d_{xw}=\text{argmin}\{d_1,d_2,\ldots,d_{nw}\}$ 对应的序列 S_{xw} 即为该异常日志流量模式 NM_w;

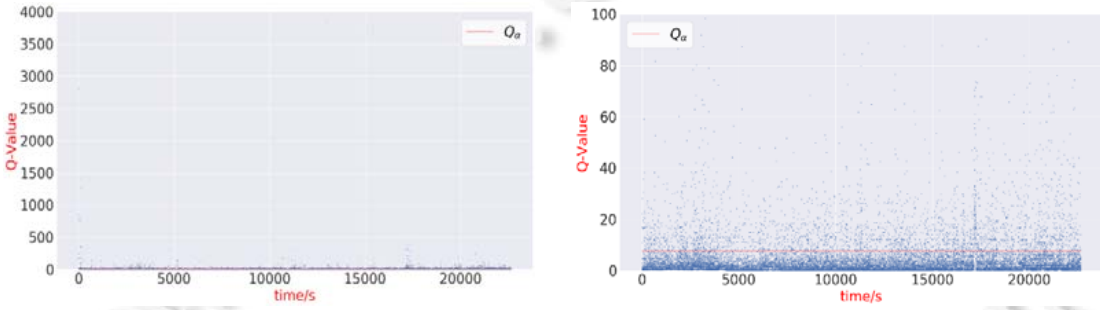
步骤 7　返回步骤 1 并重复,直至得到所有异常流量模式 $NM_1,NM_2,\ldots,NM_m$.

4　实验结果与分析评价

本节我们将第 3 节介绍的方法用于国家高性能计算环境系统在实际工作中产生的系统日志中.我们选取其系统日志的 secure 类别日志作为数据输入.在日志分类时使用了 2017 年 7 月~2017 年 12 月的日志进行分类,得到了 84 种类型的日志.考虑到日志在每天不同的时间段产生的异常日志流量模式的不同,因此我们将日志按照 3 个时间段分段,即白天、晚上、深夜(参见表 1).我们将时间片跨度设定为 5 分钟,然后按照上一小节介绍的方式得到输入矩阵并进行测试.

4.1　日志异常检测和筛选的分析评价

本小节我们使用第 3.2 节中介绍的基于主成分分析的异常检测技术对实验日志进行检测,得到各个时间片的 Q 值并与模型的阈值 Q_α进行比较,从而得到异常时间片.各个时间片的 Q 值和阈值 Q_α如图 3 所示.



(a) 整体图　　(b) 局部图

Fig.3　Q-value and threshold Q_α figure of secure logs

图 3　Secure 类型日志的白天模型 Q 值和阈值 Q_α图

从图 3 可以看出,异常类型时间片均匀地分布在整个日志周期时间片内.根据得到的异常时间片,我们可以统计正、异常时间片内不同类型日志的中位数的差异,经过实验分析,异常比正常时间片内日志类型的中位数差值为 1 进行区分,即可达到很好的过滤效果.通过该差异,我们进行了第 1 步过滤.第 2 步过滤使用基于数量的过滤,实际过滤时我们取主机阈值数 HTN=10 即可达到很好的过滤效果.根据这两条规则进行筛选后,最后得出的所有异常流量数目见表 5.

Table 5　Number of abnormal flow and its filtered

表 5　异常流量与过滤后的数目

时段	流量片段总数	异常流量片段数	异常节点流量数	过滤后异常节点流量数	压缩率(%)
白天	22 665	2 997	66 646	2 659	96.01
晚上	13 607	1 965	43 754	1 662	96.20
夜里	18 150	1 964	43 488	1 512	96.52

由表 5 可以看出:通过我们的异常检测方法和过滤规则,使得大量的流量片段数据压缩成少量的日志类型序列,大大降低了后续分析的难度.

4.2　日志层次聚类和关键类型挑选的分析评价

本节介绍由上一小节得到的大量异常日志流量序列按照第 3.3 节的方法进行层次聚类的相关实验.使用层次聚类时有两个关键参数需要定义:一是不同数据之间的距离度量方法,二是不同簇之间的距离度量方法.不同数据之间的距离定义我们按照第 3.3 节中介绍的距离公式进行计算,而不同簇间距离度量方法具有多种不同的

选择.在实验中,我们先使用不同簇间距离度量方法进行计算,然后根据结果计算其对应的共表性相关系数(cophenetic correlation coefficient)[25]来进行评价.共表相关系数越大,表明效果越好.我们使用半年 secure 类型日志的白天模型进行计算,得到不同的簇间距离计算方法对应的共表相关系数,见表 6.

Table 6 Cophenetic correlation coefficient obtained by calculating the distance between different clusters

表 6 不同簇间距离计算方法得到的共表相关系数

簇间距离计算方法	共表相关系数
平均距离标准方法(average)	0.55
最小距离标准方法(single)	0.26
最大距离标准方法(compelete)	0.50

根据表 6,我们选择使用基于平均值距离标准方法进行簇间距离的计算,因为该方法对应的共表相关系数为最大值 0.55.最终得到的层次聚类图如图 4 所示.

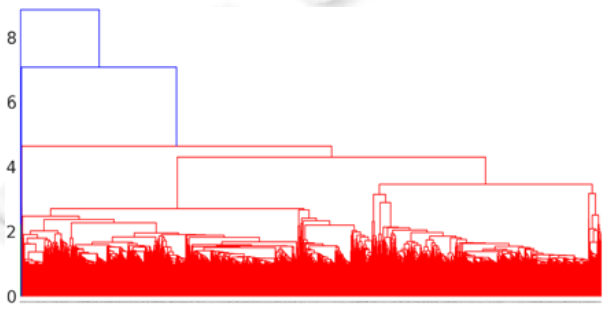


Fig.4 Hierarchical clustering of the secure logs' daytime sequence

图 4 Secure 类型日志白天序列的层次聚类图

通过图 4 可以看出:所有数据在横坐标为 1~3 内大量且迅速地聚集,之后趋于稳定.因此,选择层次聚类的距离度量的阈值大致在相对较高的位置.在实际计算时,我们选择的阈值为 3.5.使用该值进行层次聚类得到的聚类结果既可以保证类别较少,又可以使得每个小类别的聚集程度比较高.按照该阈值,我们将半年的 secure 日志早、晚、夜数据分别进行层次聚类,得到日志流量类型序列.之后使用第 3.4 节中所描述的自适应 K 项集的方法进行类别选择,得到 3 个时间区间内日志异常类型序列的代表.结果得到的前 K 项类别数量平均值以及对应的 K 值见表 7.其中,距离几何中心最近的日志异常类型序列比较长,这里就不再罗列.

Table 7 Table of secure logs' sequence

表 7 Secure 日志类型序列表

节点模式	K 值	前 K 项类型日志序列数量的几何中心
D_0	4	11:7.58;7:5.34;6:4.85;12:4.77
D_1	6	11:49.20;6:37.00;7:30.60;1:26.20;4:19.00;5:19.00
D_2	4	7:5.40;2:3.58;0:2.69;16:2.53
D_3	6	1:2.66;11:2.33;2:2.00;38:1.33;6:1.33;12:1.33
D_4	2	1:6.00;28:5.00
M_0	4	11:7.83;7:4.94;6:4.93;12:4.58;4:2.65
M_1	3	7: 4.09;2:2.61;1:2.00
M_2	4	21:3.33;1:2.00;11:1.67;20:1.33
N_0	4	11:34.80;7:27.20;3:17.80;6:14.40
N_1	4	11:6.59;7:4.22;6:3.66;12:3.64
N_2	4	7:4.95;2:3.56;0:2.44;16:2.27

由表 7 可以看出,使用半年 secure 类别日志产生的类型序列根据白天、晚上和半夜分别生成了 5、3、3 类的异常序列.我们将这 11 条异常序列继续进行层次聚类,结果如图 5 所示.根据该图可以看出:白天的模式代表

包含了大部分夜晚和半夜的模式代表,在满足将所有原本模式代表都分开的前提下,一共可分为 7 种异常日志流量模式,见表 8.

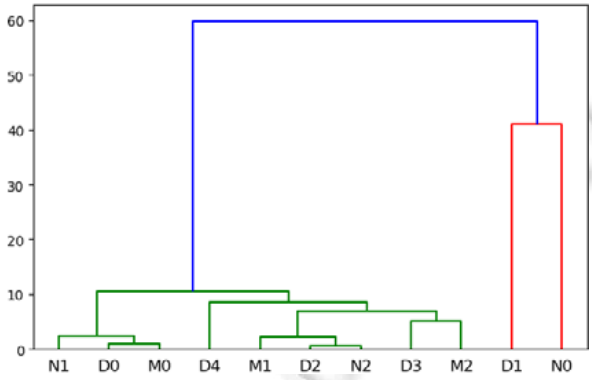


Fig.5 Hierarchical clustering of type sequences of secure log in day, night, and midnight

图 5 Secure 类型日志的白天、晚上和半夜代表类型序列的层次聚类图

Table 8 Table of sequence type of secure logs

表 8 Secure 日志序列类型表

异常流量类型序列	日志序列代表
NM_0	D_0,M_0,N_1
NM_1	D_1
NM_2	D_2,M_1,N_2
NM_3	D_3,M_2
NM_4	D_4
NM_5	N_0

我们将找出的 6 大类异常日志流量模式 NM_0、NM_1、NM_2、NM_3、NM_4、NM_5 序列对应的日志类型代表 D_0、D_1、D_2、D_3、D_4、N_0 的各个类型数量绘制的梯形图绘制出来,结果如图 6 所示.

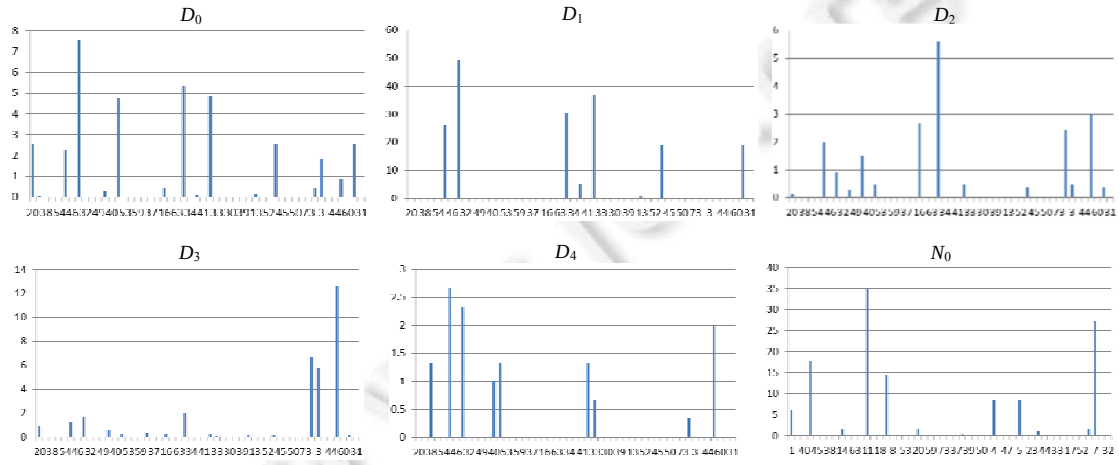


Fig.6 Type number trapezoid diagram of secure logs

图 6 Secure 型日志的类型数量梯形图

根据图示,我们可以分析以下异常流量情景.

- 情景 0:流量序列中占据异常最重要比重的日志类型是 T_{11}(failed password for invalid user 0000 from ⟨IP⟩ port ⟨PORT⟩ ssh2)和 T_7(pam_unix(sshd:auth): authentication failure).该种异常日志流量模式表明,此

时间段内可能有人进行暴力破解密码的行为;

- 情景 1:流量序列中占据异常最重要比重的日志类型是 T_{11}(同上)和 T_6(pam_unix(sshd:auth): check pass; user unknown).该种异常日志流量模式表明,此时间段内出现大量登录的行为,并且检测通过了,说明系统密码可能已经被攻破;
- 情景 2:流量序列中占据异常最重要比重的日志类型是 T_7(同上)和 T_2(pam_unix(sshd:session): session opened/closed for user).该种异常日志流量模式表明时间段内会话数量突然增多;
- 情景 3:流量序列中占据异常最重要比重的日志类型是 T_1(connection closed by ⟨IP⟩)和 T_{11}(failed password for root from ⟨IP⟩ port ⟨port⟩ ssh2).该种异常日志流量模式表明,此时间段内可能有人进行暴力破解密码的行为;
- 情景 4:流量序列中占据异常最重要比重的日志类型是 T_1(同上)和 T_{28}(usr1: TTY=pts/0; PWD=⟨PATH⟩; USER=root; COMMAND=/bin/tail -f /var/log/messages).该种异常日志流量模式表明,该时间段内用户使用的命令突然增多;
- 情景 5:流量序列中占据异常最重要比重的日志类型是 T_{11}(同上)和 T_7(同上).该种异常日志流量模式与情景 1 相同,但是出现的数量比情景 1 高出很多.

需要指出的是:由该方法得出的日志流量异常模式仅代表由机器辅助分析得出的流量异常情况,并不等同于该时间片内发生了实际的用户非法行为或系统错误.具体是否存在需要处理的异常状况,仍需要人工判断.但该方法可以自动缩小异常状况的观测范围,极大地减小了人工处理所需的工作量.

4.3 算法效率方面的分析评价

本小节统计出进行日志异常检测算法时在早、晚、深夜这 3 个不同时间段的不同操作中各个步骤所用的时间,并对结果进行分析与对比.上述讨论中,半年的日志数据在不同时间段进行检测时各个阶段所用时间详见表 9.

Table 9 Time spent in each stage of the test in different time periods

表 9 不同时间段在进行检测的各个阶段所用时间

时段	时间/占比	异常处理	异常流量筛选	层次聚类	类别选择	总时间(s)
白天	时间(s)	33.58	86.14	2041.88	39.04	2 200.64
	占比(%)	1.52	3.91	92.86	1.71	
晚上	时间(s)	18.59	50.87	656.05	24.92	750.43
	占比(%)	2.48	6.78	87.18	3.56	
夜里	时间(s)	25.15	53.05	731.75	28.15	838.10
	占比(%)	3.00	6.33	87.31	3.36	

由表 9 我们可以分析得出如下结论.

(1) 通过总时间的纵向对比我们可以发现,本文系统在处理白天产生的流量数据所用的时间明显多于晚上和夜里.这说明从总体来看,白天访问高性能计算环境的人数较多,进行的操作也较多,因此容易出现异常流量;

(2) 通过不同检测阶段的横向对比我们可以看出,层次聚类步骤消耗的时间较多.因为层次聚类算法本身的时间复杂度是 $O(n^2\log n)$(这里的 n 代表待聚类序列的数量),同时,我们选择的距离度量基于最长公共子序列算法,该算法仅需得到子序列长度,无需构造出最长公共子序列,其时间复杂度是 $O(mn)$(这里的 m 和 n 分别代表待比较的两个序列的长度).根据以上分析我们可以看出,层次聚类模块处理流量数据时受到待分析日志的数量以及每个流量序列的长度影响较大.对比白天和晚上层次聚类算法的占比也可以看出:当总时间较长时(白天>晚上),层次聚类所用时间的占比也会变大.

需要指出的是:虽然本文介绍的方法在计算时较为费时,但是该方法所进行的异常流量判别可以得到各个异常流量的情景,该步骤属于异常流量情景数据建模.当建模结束后即可得到各种异常流量情景,此时即可根据

新的流量的数字特征与已经得到的异常流量情景进行对比,从而较快速地分析出流量的异常与否以及其所属的异常流量情景.因此,后续可关注的一个研究方向是,如何将该方法得到的异常检测模型运用于实时线上的异常流量检测中.

5 总结和展望

本文介绍了一个无监督异常检测方法自动挖掘系统日志的异常日志流量模式,该检测方法可以自动找到系统日志的异常时间段,并统计时间段内不同节点的日志出现序列.本文定义了日志类型序列代表异常日志流量模式,且基于日志类型序列的相似度进行层次聚类.聚类过程中,可以全自动地得到最优参数.聚类结果根据日志类型的平均数量得到易于判断的异常日志流量模式.我们使用该系统对国家高性能计算环境下半年产生的系统 secure 类型日志进行测试,最终得到 6 种异常日志流量模式.本文在处理日志分类时,采用的是字符串比较法.所以理论上,只要是 ASCII 码格式的日志都可以自动进行分类,然后即可将单位时间的日志流量抽象成向量进行后续的异常检测处理.在实际使用中,本方法对于自然语言文本类型的日志适用性更好(该类型日志内容通常为一个英文句子,易于分类),而对于纯变量类型的日志适用性相对一般(例如 tomcat 的 access log).

本文只是针对单一日志异常流量做了一些前期探索工作,未来还有很多值得关注的研究点.今后的工作主要针对以下几个方面:将该异常模式得出的结果运用在流量的监控和预测上;该方法用于不同种类的日志,通过不同种类日志的关联关系进行分析,以找到更全面的异常日志流量模式;基于日志类型序列的角度进行更多不同维度的日志分析方法研究,例如日志类型序列的关联性分析等.

References:

[1] Zhao Y, Xiao H. The design of LARGE: Log analyzing framework in grid environment. e-Science Technology & Application, 2016, 7(3):3−7 (in Chinese with English abstract).

[2] Dokas P, Ertoz L, Kumar V, *et al*. Data mining for network intrusion detection. In: Proc. of the NSF Workshop on Next Generation Data Mining. 2002. 15.

[3] Vaarandi R. SEC—A lightweight event correlation tool. In: Proc. of the 2002 IEEE Workshop on IP Operations and Management. IEEE, 2002. 111−115.

[4] Rouillard JP. Real-Time log file analysis using the simple event correlator (SEC). In: Proc. of the Conf. on Systems Administration. DBLP, 2004. 133−150.

[5] Vaarandi R, Blumbergs B, Çalışkan E. Simple event correlator—Best practices for creating scalable configurations. In: Proc. of the IEEE Int'l Inter-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support. IEEE, 2015. 96−100.

[6] Vaarandi R. A data clustering algorithm for mining patterns from event logs. In: Proc. of the IP Operations & Management. IEEE, 2003. 119−126.

[7] Vaarandi R. A breadth-first algorithm for mining frequent patterns from event logs. In: Proc. of the IFIP Int'l Conf. on Intelligence in Communication Systems (INTELLCOMM 2004). Bangkok: DBLP, 2004. 293−308.

[8] Makanju AAO, Zincir-Heywood AN, Milios EE. Clustering event logs using iterative partitioning. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Paris: DBLP, 2009. 1255−1264.

[9] Vaarandi R, Pihelgas M. LogCluster—A data clustering and pattern mining algorithm for event logs. In: Proc. of the Int'l Conf. on Network and Service Management. IEEE, 2016. 1−7.

[10] Xu W, Ling H, Armando F, *et al*. Detecting large-scale system problems detection by mining console logs. In: Proc. of the ACM SIGOPS Symp. on Operating Systems Principles Big Sky MT. 2013. 2009.

[11] Fronza I, Sillitti A, Succi G, *et al*. Failure prediction based on log files using random indexing and support vector machines. Journal of Systems & Software, 2013,86(1):2−11.

[12] Weiss GM, Hirsh H. Learning to predict rare events in event sequences. In: Proc. of the Int'l Conf. on Knowledge Discovery & Data Mining. 1998.

[13] Yamanishi K, Maruyama Y. Dynamic syslog mining for network failure monitoring. In: Proc. of the 11th ACM SIGKDD Int'l Conf. on Knowledge Discovery in Data Mining. ACM, 2005. 499−508.

[14] Yuan D, Mai H, Xiong W, *et al*. SherLog: Error diagnosis by connecting clues from run-time logs. ACM SIGPLAN Notices, 2010, 45(3):143−154.

[15] Peng W, Li T, Ma S. Mining logs files for data-driven system management. ACM SIGKDD Explorations Newsletter, 2005,7(1): 44−51.

[16] Wang XD, Zhao YN, Xiao HL, *et al*. Research on anomaly detection system of online multi node log flow. Journal of Frontiers of Computer Science and Technology, 2020 (in Chinese with English abstract). [doi: 10.3778/j.issn.1673-9418.2002015]

[17] Hamming RW. Error detecting and error correcting codes. Bell System Technical Journal, 1950,29(2):147−160.

[18] Levenshtein VI. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, 1966,10(1): 845−848.

[19] Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 1970,48(3):443−453.

[20] Smith TF, Waterman MS. Identification of common molecular subsequences. Journal of Molecular Biology, 1981,147(1):195−197.

[21] Seker SE, Altun O, Ayan U, *et al*. A novel string distance function based on most frequent *K* characters. Int'l Journal of Machine Learning & Computing, 2014,4(2):177−183.

[22] Zhao Y, Xiao H. Extracting log patterns from system logs in LARGE. In: Proc. of the 2016 IEEE Int'l Parallel and Distributed Processing Symp. Workshops. IEEE, 2016. 1645−1652.

[23] Improvement of log pattern extracting algorithm using text similarity. In: Proc. of the Int'l Parallel and Distributed Processing Symp. Workshops. Vancouver: IEEE, 2018. 507−514.

[24] Lakhina A, Crovella M, Diot C. Diagnosing network-wide traffic anomalies. In: Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications. ACM, 2004. 219−230.

[25] Sokal RR, Rohlf FJ. The comparison of dendrograms by objective methods. Taxon, 1962,11(2):33−40.

附中文参考文献:

[1] 赵一宁,肖海力.网格环境日志分析框架 LARGE 的设计.科研信息化技术与应用,2016,7(3):3−7.

[16] 王晓东,赵一宁,肖海力,等.线上多节点日志流量异常检测系统的研究.计算机科学与探索,2020. [doi: 10.3778/j.issn.1673-9418.2002015]

王晓东(1989－),男,博士生,主要研究领域为日志处理分析,数据挖掘,机器学习.

赵一宁(1983－),男,博士,助理研究员,主要研究领域为日志处理分析,数据挖掘.

肖海力(1978－),男,博士,研究员,CCF 专业会员,主要研究领域为网格计算,分布式计算.

迟学斌(1963－),男,博士,研究员,博士生导师,CCF 杰出会员,主要研究领域为并行计算与软件,网格计算技术,高性能计算机系统维护与管理.

王小宁(1981－),女,博士,副研究员,主要研究领域为分布式计算,网格计算.