**毕业设计(论文)**

题 目**Spark框架下的网络流量异常检测与告警关联研究**

**信息安全** 系 **信息研究与安全** 专业

学员姓名 **isinstance** 班号 **nosite**

指导教员 **Hero**

2016年6月19日

**毕业设计(论文)任务书**

|  |
| --- |
| 任务来源  参加由科研项目《XXXX》研究生指出选题 |
| 任务要求  1．研究并实现基于Spark的异常流量聚类分析系统。  2．研究并实现基于Spark 的入侵检测系统日志关联分析系统。 |
| 参考文献  [1] Sandy Ryza, Uri Laserson, Sean Owen, and Josh Wills， Advanced Analytics with Spark[M] (O’Reilly). Copyright 2015 Sandy Ryza, Uri Laserson, Sean Owen, and Josh Wills, 978-1-491-91276-8:72-86.  [2] Peter Harrington， Machine Learning in Action[M]. Copyright 2012 Peter Harrington, 978-7-115-31795-7:15-27.  [39] Nick Pentreah， Machine Learn with Spark[M]. Copyright 2015 by Nick Pentreah, 978-7-115-39983-0:141-158.  [40] William Wealey， Python for Data Analysis McKinney(O’Reilly). Copyright 2013 William Wesley McKinney, 978-1-449-31979-3:115-158. |

任务起止日期：2016年4月5日至2012年6月19日

**目 录**

[摘 要 5](#_Toc7814)

[ABSTRACT 6](#_Toc5822)

[第1章 绪论 7](#_Toc19145)

[1.1 论文研究背景与意义 7](#_Toc10027)

[1.2 国内外研究现状 8](#_Toc10899)

[1.2.1 大数据处理平台研究现状 8](#_Toc25183)

[1.2.2 网络流量异常检测技术研究现状 9](#_Toc4578)

[1.2.3 网络安全告警关联技术研究现状 13](#_Toc26577)

[1.3 论文研究内容及章节安排 16](#_Toc15634)

[第2章 相关理论基础 17](#_Toc27352)

[2.1 Spark框架 17](#_Toc217)

[2.1.1 Spark框架概述 17](#_Toc10504)

[2.1.2 Spark框架的架构分析 17](#_Toc22866)

[2.1.3 Spark框架的特点及优势 18](#_Toc22433)

[2.2 k-means聚类分析方法 18](#_Toc21023)

[2.2.1 聚类分析方法概述 18](#_Toc22930)

[2.2.2 k-means聚类基本原理 21](#_Toc6374)

[2.3 Snort入侵检测系统 22](#_Toc27284)

[2.3.1 入侵检测系统概述 22](#_Toc21625)

[2.3.2 Snort入侵检测系统基本原理 26](#_Toc4140)

[2.3.3 Snort的基本组成 26](#_Toc28255)

[2.4 FP-growth频繁模式挖掘方法 27](#_Toc26508)

[2.4.1 关联分析频繁模式挖掘概述 27](#_Toc10661)

[2.4.2 FP-growth频繁模式挖掘基本原理 30](#_Toc24492)

[2.5 本章小结 31](#_Toc30995)

[第三章 Spark框架下基于k-means的流量异常检测研究与实现 32](#_Toc10522)

[3.1 Spark框架下基于k-means的流量异常检测基本思路 32](#_Toc22592)

[3.1.1 大数据处理平台选择基本思路 32](#_Toc12620)

[3.1.2 聚类方法选择基本思路 34](#_Toc17949)

[3.2 Spark框架下基于k-means的流量异常检测方法 35](#_Toc28026)

[3.3 Spark框架下基于k-means的流量异常检测技术实现 36](#_Toc13543)

[3.4 实验分析 37](#_Toc6470)

[3.4.1 实验数据集及性能指标 37](#_Toc8869)

[3.4.2 实验环境及配置 37](#_Toc28613)

[3.4.3 实验运行过程 41](#_Toc26269)

[3.4.4 结果对比分析 43](#_Toc22945)

[第4章 Spark框架下基于FP-growth的告警关联研究与实现 48](#_Toc25442)

[4.1 Spark框架下基于FP-growth的告警事件关联基本思路 48](#_Toc8551)

[4.2 Spark框架下基于FP-growth的告警事件关联方法 48](#_Toc2668)

[4.3 实验分析 50](#_Toc1866)

[4.3.1 实验数据集及性能指标 50](#_Toc17756)

[4.3.2 实验环境及配置 52](#_Toc6100)

[4.3.2 实验运行过程 53](#_Toc14927)

[4.4 本章小结 67](#_Toc1382)

[第5章 总结和展望 68](#_Toc19447)

[5.1 论文的主要工作 68](#_Toc631)

[5.2 存在的问题及工作展望 68](#_Toc31568)

[致谢 69](#_Toc19175)

[参考文献 69](#_Toc16429)

[附录 Pyspark聚类分析、关联分析代码 69](#_Toc15595)

摘 要

在APT攻击等新型复杂多步网络攻击威胁下，我国网络空间安全面临严峻挑战。利用分布式计算框架进行大数据分析是检测网络攻击的最有效途径之一。本文利用当前最适合于数据挖掘的分布式平台Spark，研究并实现了网络流量异常检测和安全事件关联分析技术。

针对网络攻击检测中流量数据的海量性问题，设计并实现了一种Spark框架下网络流量无指导异常检测系统。考虑到实时流量异常检测的需求，采用无指导快速聚类方法K-means划分流量记录所代表的攻击类型。为了避免MapReduce等传统分布式计算框架频繁硬盘读写带来的巨大时间开销，提出了一种Spark框架下的K-means检测方法，通过将每轮迭代产生的临时数据存入内存中，可有效提高K-means聚类检测算法的计算效率。在KDD99数据集上进行的实验证明该方法具有很高的检测准确性。

针对异常检测产生网络告警之后，需要通过关联告警来构建复杂多步攻击的完整攻击场景问题，提出一种Spark框架下基于FP-growth频繁模式挖掘的攻击告警关联分析方法，可有效弥补Apriori关联方法在搜索效率方面的不足。利用DARPA2000中的DDOS攻击场景重构数据集进行实验验证，结果表明XXX方法的有效性。

**关键词：**攻击检测 数据挖掘 FP-growth关联分析 k-means聚类分析

# ABSTRACT

With the growing demand of modern social information security, network and people's life more and more closely related, network to people brings convenience, is inevitably accompanied by risks, the risk is that the network attacks.

Network attacks, the use of computer networks to the other side of the computer and network attacks, to achieve their own purposes. And corresponding is to invade detection system, intrusion detection system can quickly find network attack in the event and the network attacks responded quickly, make up for the loopholes in security, for individual users, a local computer can do intrusion detection and analysis of the dual effect, but for large Internet companies and websites, without a strong computing ability and rapid response capability is unable to cope with the increasingly serious rampant cyber attacks. It is a good method with apt attack technology matures and development. Facing the long time apt attacks, how to reverse the situation, for the site's operation and maintenance team, the moment to moment flow were recorded and analyzed. Web Intrusion Detection System of recorded data is analyzed and classified, judge each flow is likely to become dangerous intruders before the Horse Artillery, how the apt attacks at the beginning or haven't caused massive damage to do prediction attacks, and improve the security level, bug fixes, nowhere to hide the APT attackers.

**KEYWORDS:** Information safety，Data mining，Machine learning

1. 绪论

**1.1 论文研究背景与意义**

近年来，APT攻击等复杂多步攻击对网络安全的危害越来越大，因此日益受到网络安全工作人员的关注[1][2]。APT攻击是通过对目标进行长时间的监测而实施大范围部署的攻击方式。由于这种攻击方式的不对称性和隐蔽性，给安全人员的检测和防护带来了诸多困难。面对严峻的APT攻击防御形势，提前预警攻击行为势在必行，而大数据分析技术是实现攻击预警的最有效途径之一。大数据分析技术可以利用在多台计算机之间形成的计算群来处理海量数据，通过在一堆无序的网络数据中发现规律，大数据分析能较好的预测攻击行为[3]。此外，大数据分析技术具有计算可扩展、计算可广播、不受地域影响、数据特征和关系的探索等优点，使其成为网络攻击安全检测的重要工具。

要想大幅度减少APT攻击对互联网企业安全的影响[4][5]，提高互联网安全水平，可以采取如下防范措施：一是健全互联网企业的安全管理，提升安全人员处理网络攻击时的反应速度，从而可以在攻击发生之后迅速组织力量，保护企业的信息安全，如可向专门的安全公司提出保护的申请，或者聘用网络安全人员进驻公司。二是通过大数据分析技术对网络流量进行处理和分析，预测网络攻击发生的时间和攻击类型，进而采取有效的防范措施加以防范。网络攻击的发起并不是一瞬间的，网络攻击者对目标进行攻击之前，对目标进行全面的扫描和评估，商讨使用的攻击方法，进行攻击前的准备，然后才实施攻击。针对网络攻击者的先前扫描流量的捕获和分析，可以预测网络攻击的发生。相对于第一种亡羊补牢的方法，第二种方法更具有现实意义。

文章拟通过补充分析当前开源世界所拥有的大数据计算相关技术，重点讨论Spark的工作原理和几种典型的数据挖掘算法。在深入分析Spark框架下数据挖掘算法的工作原理后，根据机器学习原理设计并实现了基于K-Means的异常流量检测系统和基于FP-growth的网络告警关联分析系统。最后根据设计的测试方案对流量数据和告警数据进行分析和预测。研究工作分析了网络异常流量检测技术，防范APT攻击具有较好的理论意义实际应用价值。

**1.2 国内外研究现状**

面对APT攻击等复杂多步网络攻击，国内外研究人员提出了很多有效的防御技术和方法，这些方法各有优点和适用范围。本文重点研究基于大数据处理平台的异常流量检测和安全事件关联分析，下面从大数据处理平台、异常流量检测和告警关联分析三个方面对国内外研究现状进行简要综述。

**1.2.1 大数据处理平台研究现状**

为了满足日益增长的大数据分析需求，研究者提出了许多用于大数据分析的开源大数据计算模型。这些方案主要基于两种不同的基础原型：Hadoop和Spark。

Hadoop由Apache Software Foundation 公司于2005年秋天作为[Lucene](http://baike.baidu.com/view/371811.htm)的子项目[Nutch](http://baike.baidu.com/view/46642.htm)的一部分正式引入。它受到最先由 GoogleLab开发的Map/Reduce和GoogleFile System([GFS](http://baike.baidu.com/view/805525.htm))的启发。2006年3月份，Map/Reduce和Nutch Distributed File System (NDFS)分别被纳入称为Hadoop的项目中。

Hadoop原本来自于谷歌一款名为MapReduce的编程模型包。谷歌的MapReduce[5]框架可以把一个应用程序分解为许多并行计算指令，跨大量的计算节点运行非常巨大的数据集。使用该框架的一个典型例子就是在网络数据上运行的搜索算法。Hadoop最初只与网页索引有关，迅速发展成为分析大数据的领先平台。

Spark由加州大学伯克利分校AMP实验室(Algorithms, Machines, and People Lab) 开发，可用来构建大型的、低延迟的数据分析应用程序。Spark是在[Scala](http://baike.baidu.com/view/1588150.htm)语言中实现的，它将Scala用作其应用程序框架。与 Hadoop不同，Spark和Scala能够紧密集成，其中的Scala可以像操作本地集合对象一样轻松地操作分布式数据集。

尽管创建 Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对Hadoop的补充，可以在 Hadoop文件系统中并行运行。通过名为 Mesos的第三方集群框架可以支持此行为。Spark设计理念与核心基于RDD（Resilient Distributed Dataset，弹性分布式数据），可以提供一站式多维度的大数据计算模型，并在同一个技术堆栈中快速对数据集进行批处理、即席查询、机器学习、图计算和准实时流处理等。

**1.2.2 网络流量异常检测技术研究现状**

网络行为学认为网络的流量行为具有长期特征和短期特征。网络长期特征表现在网络行为具有一定的规律性和稳定性[7][8]。能够对局域网的流量或者某些关键主机的流量情况进行实时监测，并进行预测与分析，有助于判定异常网络流量，及早发现和识别潜在的入侵攻击的发生。网络流量异常会严重影响网络性能，造成网络拥塞，严重的会使网络中断，使网络设备利用率达到100%，无法响应进一步指令。造成网络异常流量的原因可能有:网络扫描、DDos攻击、蠕虫、恶意下载、用户对网络资源的不当使用以及物理链路或者设备不能正常运转等[9][10]。基于网络流量的入侵检测是识别入侵的一个新的研究方向，特别针对网络漏洞扫描、 DDoS攻击以及蠕虫病毒等较为有效[11][12]。入侵检测中通常用到的异常检测方法有：统计方法(Staticstics)、神经网络(Neural network)、模式预测( Patten prediction)、遗传算法(Genetic Algorithm)、序列匹配与学习 (Sequence matching and learning)、免疫学方法(Immunology)、数据挖掘(Data mining)等等。目前出现的基于网络流量分析的入侵检测技术如下：

（1）统计模型方法

统计方法通过分析大量的系统参数生成系统的正常行为轮廓库，并自适应地学习系统的正常行为模式。每个行为轮廓代表一个主体的正常行为，由一组入侵检测度量值来描述。主体的审计记录由一组入侵检测测量值来表示，并与相关的行为轮廓进行比较，如果两者确定的N维空间的点相差足够远，则被视为异常。Denning描述了5个经典统计模型：操作模型、平均值和标准差模型、多元模型、马尔可夫过程模型和时序模型。 统计方法的最大优点是它可以“学习”用户的行为习惯，从而具有较高的检测率与可用性。通过对一段时间网络安全状况下数据包流量进行统计，可从宏观上建立起网络在安全状况下的周期数据包流量轮廓，用于检测以拒绝服务攻击为主的入侵网络扫描等异常行为。 统计方法可以利用成熟的统计理论成果，但它对事件发生的次序及内在联系等不敏感、检测系统可能被入侵者逐渐训练而将入侵视为正常行为。

（2）小波分析与自相似特征方法

小波分析是近十几年才发展起来并迅速应用到信号处理和图像处理等众多领域的一种数学工具。它具有多分辨率分析的特点，而且在时-频两域都具有表征信号局部特征的能力，是一种窗口大小固定不变但其形状可改变、时间窗和频率窗都可以改变的时频局部化分析方法。即在低频部分具有较高的频率分辨率和较低的时间分辨率，在高频部分具有较高的时间分辨率和较低的频率分辨率，很适合于探测正常信号中夹带的瞬态反常现象并展示其成分，所以被誉为分析信号的显微镜。小波变换能有效地从信号中提取信息,通过伸缩和平移等运算功能对函数或信号进行多尺度分析(Multi-resolution Analysis)。 小波变换可得到低频系数(或叫作近似系数)和高频系数(或叫作细节系数)，其中低频系数反映原始信号的轮廓，而高频系数 则是反映信号的细节。根据数字信号理论可知，信号中的奇异性往往是由于频率突变造成，也就是说，这种奇异性一般是通过频率的异常变化而反映出来，那么只要把信号中的频率变化情况提取出来，就可以发现信号中奇异性，并确定奇异性发生的时间。

自相似性是指网络的负载随着时间的扩展常常表现出自相似的模式。1994年，Leland等人对Bellcore的局域网测试与分析的结果显示，实际网络流量模型具有统计自相似性。 1995年，Beran等通过对大量的不同类别的可变比特率视频流数据的统计发现，它们也同样表现出一种长相关特性。另外，WAN、FASTPAC等网络的测量，同样发现这些网络业务量表现出长相关的特性。对于网络流量的自相似特征的研究还有很多的工作，而且经常将他们和小波分析联系起来研究。小波分析能将复杂的非线性网络流量时间序列分解成不同频率的子序列。基于小波分解的思想，利用网络流量的自相似特性来对网络的异常行为进行检测，可了根据网络流量自相似特征参数的偏差来给出检测攻击的方法。首先在小波域内对网络流量进行分解，计算表征网络流量突发性的重要参数(Hurst指数)，然后根据正常网络流量Hurst指数和异常网络流量Hurst指数的偏差来检测攻击，对不同时间尺度下Hurst指数的变化进行了比较分析。Narasimba Reddy等提出对长时稳定流的流量进行小波变换的方法发现DDoS异常。该方法利用小波变换工具放大异常信号，提高了灵敏度，但算法实现复杂，且不能定位异常点。

（3）CUSUM方法

CUSUM(Cumulative Sum)算法是在统计过程控制中常用的算法，它可以检测到一个统计过程均值的变化。CUSUM基于这一事实：如果有变化发生，随机序列的概率分布也会改变。通常，CUSUM 需要随机序列的参数模型，以便可以用概率密度函数来监控序列。不幸的是，计算机网络是一个非常动态而复杂的实体，因特网业务模型的理论结构是一个复杂的问题，因而，一个主要的难题是如何模拟随机序列{Xn}。而非参数方法不需要具体的模型，更适合于分析因特网。非参数CUSUM算法提出的主要思想是：累积明显比正常运行情况下平均水平高的Xn的值。这一算法的优点之一就是能以连续方式监控输入随机变量，从而达到实时检测的目的。

（4）数据挖掘方法

数据挖掘(Data mining)是一种特定应用的数据分析过程， 可以从包含大量冗余信息的数据中提取尽可多的蕴藏的安全信息，抽象出有利于进行判断和比较的特征模型。这些特征模型可以是基于异常检测的特征量模型，也可以是基于异常检测的行为描述模型。根据这些特征向量模型和行为描述模型，可以由计算机利用相应的算法判断出当前行为的性质。 在众多的数据挖掘方法中有几类方法对入侵检测非常有用。比如说数据分类的方法，关联分析的方法以及序列分析的方法等等。

但是，由于数据挖掘通常只能对离散值进行处理，在数据预处理中要将连续属性域划分为若干离散区间，这就导致了所谓的“尖锐边界”(sharp boundary)问题。为了解决这一问题，国内外一些学者提出了结合模糊集理论的模糊数据挖掘技术，并取得了很好的效果。数据挖掘技术的优点是可以处理大量数据的情况，但对于入侵检测来说，得到足够的样本是非常困难和昂贵的。

（5）神经网络方法

神经网络是一个由简单处理元构成的规模宏大的并行分布式处理器。天然具有存储经验知识和使之可用的特性。由于神经网络适合于学习比较复杂的非线性关系，而且它是数据驱动不断学习的，即通过学习掌握数据间的依从关系，不需要对网络流量进行大量的数学建模工作。所以，适合解决网络流量模式中的复杂相关性和预测中的自适应问题。在现有的神经网络模型中，采用较多的是多层前馈结构，尤其是三层神经网络结构。根据不同的预测需要，可以利用单个输出单元预测下时刻的流量，也可以利用多输出单元预测下多个时刻流量或者下一时刻流量。常用的算法通常是反向传播(BP)算法。对于BP神经网络来说，隐藏节点的增多可能导致过度的问题，而过度拟合会损坏网络的学习能力，针对只运用BP神经网络作网络流量预测的局限性，各种改进的应用于网络流量预测的神经网络模型不断被引入：模糊理论与神经网络结合的模糊神经网络、信号处理中的FIR数字滤波器与神经网络结合的FIR神经网络、将时间引入神经网络的时延回归神经网络等。

（6）其他方法

基于网络流量异常的入侵检测系统，仍然处于研究的初级阶段，还有其他一些方法如免疫机理方法、自适应Ａgent、 遗传算法等等。目前应用到商用产品上的，大多只是集成在防火墙上的一些简单规则或初级的入侵检测系统功能。而这些方法大都还停留在理论研究阶段，离实际应用还有一段路要走。

**1.2.3 网络安全告警关联技术研究现状**

安全事件关联分析技术通过对安全事件进行关联分析，生成更高级的攻击场景[13][14]。为了便于有效处理安全事件信息，很多学者进行了有益的研究，系统的描述了报警信息的标准化问题[15][16]。并且实现了Snort、BlackIcc 两个系统的报警格式标准化。报警格式主要是遵照了IDMEF格式，其中比较关键的报警名称统一是通过手工建立报警名称与CVE号的对应关系库实现。实现报警格式标准化之后，根据IDMEF标准以及安全事件关联分析的需要，可以适当扩展数据，即对数据进行预处理操作。预处理组件(Pre—Processing)，主要完成对扩展报警信息的信息内容完善。

安全事件关联分析技术可以分为三类：基于相似概率的关联算法、基于情景的关联算法、基于安全事件前因与后果的关联算法等。

1. 基于相似概率的关联

通常认为相关告警具有一定的属性相似性[18]。此假设可以从实际攻击过程的分析中得到验证。基于此种经验假设，一些学者对报警之间的相似性进行了深入的研究。 SRI的Andersson等和Valdes、Skinerza在EMERALD项目中提出了基于告警属性相似度的安全报警关联系统，利用手工定义的入侵事件间概率 相似度和极小匹配规则来构建安全事件关联分析系统。Stanifor等提出的Spice系统和Julisch同样用到了相似度计算的方法。Cuppens等在法国国防部(DGA：French Defense Agency)的科研项目MIRADOR中的alert clustering部分使用了类似的聚类方法。这类方法的优点在于对相似报警(来自相同的源或目标地址的报警)进行聚类时非常有效，缺点是不能充分发掘出相关报警之间的因果关系。Debar和Wespi提出的用聚类和关联两种方法用于IDS报警的关联分析，综合考虑了相似性和因果两种关系，部分解决了该问题。OliverM．Dain等提出的基于自定义的报警间距离的场景构建方法，用概率的方法定义了报警间距离，然后将相近的报警聚成一簇[19]。

（2）基于攻击情景的关联

基于攻击情景的关联将入侵过程描述为攻击情景，攻击情景一般可表示为五个步骤或阶段的组合：信息探测、攻击尝试、权限提升、深入攻击、拒绝服务。但实际上攻击情景是由人或者通过机器学习得到的经常发生的攻击序列，这种关联方法类似于入侵检测中误用检测的方法。Dain提出通过机器学习的方法来训练包含已知入侵情景的数据集来“学习”告警关联模型。这种方法能实时地将告警聚合到“情景”中(每个情景是由同一入侵者发起的一系列的攻击而引起的告警组合)。它根据计算一个新告警属于给定情景的概率来自动建立报警关联模型。但是他需要在每一个配置中进行训练，而且结果模型可能与训练数据匹配太紧密，于是就会漏掉没有出现在训练数据集中的攻击情景。

1. 基于安全事件因果关系的关联

使用攻击前提条件和后续结果来关联告警的方法主要是针对攻击者发动攻击的前因后果而设计的。该算法的基本思想是：寻找一个攻击事件A发起的先决条件和攻击事件B的攻击结果之间是否存在逻辑联系，如果存在，则A和B就可能是一个系列攻击的两个环节，可以关联为B—A。 Ning在他的TIAA系统中事先定义了各种攻击可能发生的前因和后果的知识库，通过对报警实例之间前因和后果的匹配，形成告警关联图。该算法有三个比较关键的步骤：第一步是要提前定义每一种已知攻击类型的先决条件和攻击后果。第二步是根据时间限制和hyper-alert type定义来生成 hyper-alert实例，生成hyper-alert实例的过程实际上也是对原始报警信息的一个聚合过程。第三步是对生成好的hyper-alert实例进行关联的过程。有了对攻击的先决条件和攻击后果的详细描述，关联过程就相对容易的多。实验证明，这种方法可以有效的发现报警之间的因果关系，但关联的效果依赖于知识库的制定，新的攻击层出不穷，定义一套系统的、完善的前因后果知识库不太可行，并且对于每个安全事件都要进行前因与后果的关联分析，以及归属决策，需要耗费大量的计算机资源，不利于实时在线处理。

**1.3 论文研究内容及章节安排**

本文在介绍数据挖掘工作原理和聚类、关联分析方法的基础上，实现了一种基于Spark的入侵检测系统。通过本文的研究，有效解决了入侵检测分析过程中存在的主观随意性和思维不定性问题。本文主要可研究内容，即章节安排如下：

第一章，绪论。主要阐述本文的研究背景和意义，介绍网络流量异常检测技术与网络安全告警关联技术的研究现状。

第二章，相关理论基础。简要阐述Spark的基本原理和工作方法，并对几种典型的大数据平台、聚类分析方法、入侵检测系统、关联分析方法进行系统分析和比较。

第三章，Spark框架下基于K-means的流量异常检测研究与实现。选择有效的方法对KDD99数据进行量化和预处理，构建基于K-means的流量异常检测体系，设定不同K值，从而建立流量异常检测系统。

第四章，Spark框架下基于FP-growth的告警事件关联研究与实现。结合林肯实验室DPRAPA流量数据案例，给出基于FP-growth下告警事件分析详细过程。

第五章，总结与展望。总结基于Spark的入侵检测系统的分析思想与方法，并对该系统的改进措施提出合理的构想。

**第2章 相关理论基础**

**2.1 Spark框架**

**2.1.1 Spark框架概述**

Spark是近年来发展较快的分布式并行数据处理框架，可与Hadoop联合使用，以增强Hadoop的性能。同时，Spark还增加了内存缓存、流数据处理、图数据处理等更为高级的数据处理能力。Spark自从2009年开源其代码之后，获得了长足的发展。目前是开源基金支持的三大开源项目之一。

**2.1.2 Spark框架的架构分析**

Spark的运行模式多种多样，灵活多变，部署在单机上时，既可以用本地模式运行，也可以用伪分布式模式运行；而当以分布式集群的方式部署时，有众多的运行模式可供选择，这取决于集群的实际情况，底层的资源调度既可以依赖于外部的资源调度框架，也可以使用Spark内建的Standalone模式。对于外部资源调度框架的支持，目前的实现包括相对稳定的Mesos模式，以及还在持续开发更新中的Hadoop YARN模式。

Spark基本工作流程：



图1 Spark基本工作流程

从图1中可以看到，所有的Spark应用程序都离不开SparkContext和Executor两部分，Executor负责执行任务，运行Executor的机器称为Worker节点，SparkContext由用户程序启动，通过资源调度模块和Executor通信。SparkContext和Executor这两部分的核心代码实现在各种运行模式中都是公用的。在它们之上，根据运行部署模式的不同，包装了不同调度模块以及相关的适配代码。

以SparkContext为程序运行的总入口，在SparkContext的初始化过程中，Spark会分别创建DAGScheduler作业调度和TaskScheduler任务调度两级调度模块。

其中作业调度模块是基于任务阶段的高层调度模块，它为每个Spark作业计算具有依赖关系的多个调度阶段（通常根据shuffle来划分），然后为每个阶段构建出一组具体的任务（通常会考虑数据的本地性等），然后以TaskSets（任务组）的形式提交给任务调度模块来具体执行。而任务调度模块则负责具体启动任务、监控和汇报任务运行情况。

**2.1.3 Spark框架的特点及优势**

与Hadoop相比，为了在快速的处理大量数据上得到满意的速度，Spark 启用了内存分布数据集，除了能够提供交互式查询外，它还可以优化迭代工作负载。Spark拥有Hadoop MapReduce所具有的优点；但不同于MapReduce的是，Job中间输出结果可以保存在内存中，从而不再需要读写HDFS，因此Spark能更好地适用于数据挖掘与机器学习等需要迭代的MapReduce的算法。

**2.2 k-means聚类分析方法**

**2.2.1 聚类分析方法概述**

聚类分析法是理想的多变量统计技术，主要有分层聚类法和迭代聚类法。 聚类分析也称群分析、点群分析，是研究分类的一种多元统计方法。

基本思想：样品（网点）或指标（变量）之间存在程度不同的相似性（亲疏关系——以样品间距离衡量）。于是根据一批样品的多个观测指标，具体找出一些能够度量样品或指标之间相似程度的统计量， 以这些统计量为划分类型的依据。把一些相似程度较大的样品（或指标）聚合为一类，把另外一些彼此之间相似程度较大的样品（或指标）又聚合为另一类，直到把所有的样品（或指标）聚合完毕，这就是分类的基本思想。在聚类分析中，通常我们将根据分类对象的不同分为Q型聚类分析和R型聚类分析两大类。

现有的聚类算法多种多样。而对于具体应用，聚类算法的选择取决于数据的类型、聚类的目的。如果聚类分析被用作描述或探查的工具，可以对同样的数据尝试多种算法，以发现数据可能揭示的结果。

主要的聚类算法有如下几类：划分方法、层次方法、基于密度的方法、基于网格的方法以及基于模型的方法。

每一类中都存在着得到广泛应用的算法，例如：划分方法中的K-Means聚类算法、层次方法中的凝聚型层次聚类算法、基于模型方法中的神经网络聚类算法等。

目前,聚类问题的研究不仅仅局限于上述的硬聚类，即每一个数据只能被归为一类，模糊聚类也是聚类分析中研究较为广泛的一个分支。模糊聚类通过隶 属函数来确定每个数据隶属于各个簇的程度，而不是将一个数据对象硬性地归类到某一簇中。目前已有很多关于模糊聚类的算法被提出，如著名的FCM算法等。

与其他聚类算法相比，K-Means聚类算法具有算法简单、计算快速等优点。但是也存在缺点，如算法初始值的选取比较复杂，很多时候K值的选取花的时间比K-Means聚类花的时间还多。

依据层次分解的顺序是自底向上的还是自上向下的，层次聚类算法可分为凝聚的层次聚类算法和分裂的层次聚类算法。凝聚型层次聚类的策略是先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到所有对象都在一个簇中，或者某个终结条件被满足。绝大多数层次聚类属于凝聚型层次聚类，它们只是在簇间相似度的定义上有所不同。采用最小距离的凝聚层次聚类算法流程：

(1) 将每个对象看作一类，计算两两之间的最小距离；

(2) 将距离最小的两个类合并成一个新类；

(3) 重新计算新类与所有类之间的距离；

(4) 重复(2)、(3)，直到所有类最后合并成一类。

SOM神经网络是由芬兰神经网络专家Kohonen教授提出的，该算法假设在输入对象中存在一些拓扑结构或顺序，可以实现从输入空间(n维)到输出平面(2维)的降维映射，其映射具有拓扑特征保持性质,与实际的大脑处理有很强的理论联系。SOM网络包含输入层和输出层。输入层对应一个高维的输入向量，输出层由一系列组织在2维网格上的有序节点构成，输入节点与输出节点通过权重向量连接。学习过程中，找到与之距离最短的输出层单元，即获胜单元，对其更新。同时，将邻近区域的权值更新，使输出节点保持输入向量的拓扑特征。算法流程：

(1) 网络初始化，对输出层每个节点权重赋初值；

(2) 将输入样本中随机选取输入向量，找到与输入向量距离最小的权重向量；

(3) 定义获胜单元，在获胜单元的邻近区域调整权重使其向输入向量靠拢；

(4) 提供新样本、进行训练；

(5) 收缩邻域半径、减小学习率、重复，直到小于允许值，输出聚类结果。

1965年美国加州大学柏克莱分校的扎德教授第一次提出了‘集合’的概念。经过十多年的发展，模糊集合理论渐渐被应用到各个实际应用方面。为克服非此即彼的分类缺点，出现了以模糊集合论为数学基础的聚类分析。用模糊数学的方法进行聚类分析，就是模糊聚类分析。FCM算法是一种以隶属度来确定每个数据点属于某个聚类程度的算法。该聚类算法是传统硬聚类算法的一种改进。算法流程：

(1) 标准化数据矩阵；

(2) 建立模糊相似矩阵，初始化隶属矩阵；

(3) 算法开始迭代，直到目标函数收敛到极小值；

(4) 根据迭代结果，由最后的隶属矩阵确定数据所属的类，显示最后的聚类结果。

**2.2.2 k-means聚类基本原理**

输入：聚类个数k，以及包含 n个数据对象的数据库。

输出：满足方差最小标准的k个聚类。

k-means 算法接受输入量 k ；然后将n个数据对象划分为 k个聚类以便使得所获得的聚类满足：同一聚类中的对象相似度较高；而不同聚类中的对象相似度较小。聚类相似度是利用各聚类中对象的均值所获得一个“中心对象”（引力中心）来进行计算的[20]。  
 k-means 算法的工作过程说明如下：首先从n个数据对象任意选择 k 个对象作为初始聚类中心；而对于所剩下其它对象，则根据它们与这些聚类中心的相似度（距离），分别将它们分配给与其最相似的（聚类中心所代表的）聚类；然 后再计算每个所获新聚类的聚类中心（该聚类中所有对象的均值）；不断重复这一过程直到标准测度函数开始收敛为止。一般都采用均方差作为标准测度函数. k个聚类具有以下特点：各聚类本身尽可能的紧凑，而各聚类之间尽可能的分开[21][22]。

**2.3 Snort入侵检测系统**

**2.3.1 入侵检测系统概述**

[入侵检测](http://baike.baidu.com/view/16487.htm)系统（intrusion detection system，简称IDS）是一种对网络传输进行即时监视，在发现可疑传输时发出警报或者采取主动反应措施的网络安全设备。它与其他网络安全设备的不同之处在于，IDS是一种积极主动的安全防护技术。IDS最早出现在1980年4月。1980年代中期，IDS逐渐发展成为入侵检测专家系统（IDES）。1990年，IDS分化为基于网络的IDS和基于[主机](http://baike.baidu.com/view/23880.htm)的IDS。后又出现分布式IDS。目前，IDS发展迅速，已有人宣称IDS可以完全取代[防火墙](http://baike.baidu.com/view/3067.htm)。

IDS是计算机监视系统，它通过实时监视系统，一旦发现异常情况就发出警告。IDS入侵检测系统以信息来源的不同和检测方法的差异分为几类：根据信息来源可分为基于主机IDS和基于网络的IDS，根据检测方法又可分为异常入侵检测和误用入侵检测。不同于防火墙，IDS入侵检测系统是一个监听设备，没有跨接在任何链路上，无须网络流量流经它便可以工作。因此，对IDS的部署，唯一的要求是：IDS应当挂接在所有所关注流量都必须流经的链路上。在这里，"所关注流量"指的是来自高危网络区域的访问流量和需要进行统计、监视的网络报文。在如今的网络拓扑中，已经很难找到以前的HUB式的共享介质冲突域的网络，绝大部分的网络区域都已经全面升级到交换式的网络结构。因此，IDS在交换式网络中的位置一般选择在尽可能靠近攻击源或者尽可能靠近受保护资源的位置。这些位置通常是：服务器区域的交换机上；Internet接入路由器之后的第一台交换机上；重点保护网段的局域网交换机上。由于入侵检测系统的市场在近几年中飞速发展，许多公司投入到这一领域上来。Venustech(启明星辰）、Internet Security System（ISS）、思科、赛门铁克等公司都推出了自己的产品。

入侵检测系统分为四个组件：

（1）事件产生器（Event generators），它的目的是从整个计算环境中获得事件，并向系统的其他部分提供此事件。

（2）事件分析器（Event analyzers），它经过分析得到数据，并产生分析结果。

（3）响应单元（Response units），它是对分析结果作出反应的功能单元，它可以作出切断连接、改变文件属性等强烈反应，也可以只是简单的报警。

（4）事件数据库（Event databases），事件数据库是存放各种中间和最终数据的地方统称，它可以是复杂的数据库，也可以是简单的文本文件。

入侵检测系统的弱点主要为数据的检测受到攻击时对自身防护的影响。由于现在互联网发展迅猛，网络传输速率大大加快，这加大了IDS工作时的负荷，也意味着IDS对攻击活动的检测可靠性不高。而IDS在处理指向自己的攻击时，对其他连接的检测将会被抑制。同时由于模式识别技术的不完善，IDS的高虚警率也是它的一大问题。

入侵检测系统根据入侵检测的行为分为两种模式：异常检测和误用检测。前者先要建立一个系统访问正常行为的模型，凡是访问者不符合这个模型的行为将被断定为入侵；后者则相反，先要将所有可能发生的不利的、不可接受的行为归纳建立一个模型，凡是访问者符合这个模型的行为将被断定为入侵。

这两种模式的安全策略是完全不同的，而且，它们各有长处和不足：异常检测的漏报率很低，但是不符合正常行为模式的行为并不见得就是恶意攻击，因此这种策略误报率较高；误用检测由于直接匹配比对异常的不可接受的行为模式，因此误报率较低。但恶意行为千变万化，可能没有被收集在行为模式库中，因此漏报率就很高。这就要求用户必须根据本系统的特点和安全要求来制定策略，选择行为检测模式。现在用户都采取两种模式相结合的策略。

在异常入侵检测系统中常常采用的检测方法有以下几种。

（1）基于贝叶斯推理检测法：是通过在任何给定的时刻，测量变量值，推理判断系统是否发生入侵事件。基于特征选择检测法：指从一组度量中挑选出能检测入侵的度量，用它来对入侵行为进行预测或分类。基于贝叶斯网络检测法：用图形方式表示随机变量之间的关系。通过指定的与邻接节点相关一个小的概率集来计算随机变量的联接概率分布。按给定全部节点组合，所有根节点的先验概率和非根节点概率构成这个集。贝叶斯网络是一个有向图，弧表示父、子结点之间的依赖关系。当随机变量的值变为已知时，就允许将它吸收为证据，为其他的剩余随机变量条件值判断提供计算框架。

（2）基于模式预测的检测法：事件序列不是随机发生的而是遵循某种可辨别的模式是基于模式预测的异常检测法的假设条件，其特点是事件序列及相互联系被考虑到了，只关心少数相关安全事件是该检测法的最大优点。

（3）基于统计的异常检测法：是根据用户对象的活动为每个用户都建立一个特征轮廓表，通过对当前特征与以前已经建立的特征进行比较，来判断当前行为的异常性。用户特征轮廓表要根据审计记录情况不断更新，其保护去多衡量指标，这些指标值要根据经验值或一段时间内的统计而得到。

（4）基于机器学习检测法：是根据离散数据临时序列学习获得网络、系统和个体的行为特征，并提出了一个实例学习法IBL，IBL是基于相似度的方法，该方法通过新的序列相似度计算将原始数据（如离散事件流和无序的记录）转化成可度量的空间。然后，应用IBL学习技术和一种新的基于序列的分类方法，发现异常类型事件，从而检测入侵行为。其中，成员分类的概率由阈值的选取来决定。

（5）数据挖掘检测法：数据挖掘的目的是要从海量的数据中提取出有用的数据信息。网络中会有大量的审计记录存在，审计记录大多都是以文件形式存放的。如果靠手工方法来发现记录中的异常现象是远远不够的，所以将数据挖掘技术应用于入侵检测中，可以从审计数据中提取有用的知识，然后用这些知识区检测异常入侵和已知的入侵。采用的方法有KDD（Knowledge Discovery in Database）算法，其优点是善于处理大量数据的能力与数据关联分析的能力，但是实时性较差。

（6）基于应用模式的异常检测法：该方法是根据服务请求类型、服务请求长度、服务请求包大小分布计算网络服务的异常值。通过实时计算的异常值和所训练的阈值比较，从而发现异常行为。

（7）基于文本分类的异常检测法：该方法是将系统产生的进程调用集合转换为“文档”。利用K邻聚类文本分类算法，计算文档的相似性。

误用入侵检测系统中常用的检测方法有以下几种。

（1）模式匹配法：是常常被用于入侵检测技术中。它是通过把收集到的信息与网络入侵和系统误用模式数据库中的已知信息进行比较，从而对违背安全策略的行为进行发现。模式匹配法可以显著地减少系统负担，有较高的检测率和准确率。

（2）专家系统法：这个方法的思想是把安全专家的知识表示成规则知识库，再用推理算法检测入侵。主要是针对有特征的入侵行为。

（3）基于状态转移分析的检测法：该方法的基本思想是将攻击看成一个连续的、分步骤的并且各个步骤之间有一定的关联的过程。在网络中发生入侵时及时阻断入侵行为，防止可能还会进一步发生的类似攻击行为。在状态转移分析方法中，一个渗透过程可以看作是由攻击者做出的一系列的行为而导致系统从某个初始状态变为最终某个被危害的状态。

**2.3.2 Snort入侵检测系统基本原理**

Snort是由一个简单的网络管理工具发展而成的分布式入侵检测系统，Snort可以用作嗅探器、包记录器或者网络入侵检测系统NIDS。作嗅探器时，Snort将对发往同一个网络其他主机的流量进行捕获和显示。嗅探器利用了局域网内的流量包共享的漏洞，它可以将局域网路上传输的每一个数据包的内容都显示在你的屏幕上，包括数据包头和数据包的载荷。以数据包记录器模式运行时，Snort以和嗅探器相似的方式抓取数据包，不同的是将收集的数据记入日志而不是显示在屏幕上。当Snort以网络入侵检测系统（NIDS）模式运行时，Snort也抓取并存储网络上传输的每一个包，不同在于NIDS模式能对数据包内的载荷进行分析和处理，这种处理不是简单的将数据写入文件或是显示在屏幕上，而是对每一个数据包的载荷进行检查以判断这个数据包的本质是无恶意的还是有恶意的。当发现可疑的流量时，Snort就会告警。NIDS因其能监控大片网段而比其他类型的IDS更受欢迎，在这里要研究的是NIDS模式下的Snort。

**2.3.3 Snort的基本组成**

Snort有5个主要模块：libpcap模块、解码器模块、预处理程序模块、检索引擎模块、输出模块。libpcap模块把数据包以原始状态抓取后送到解码器模块。解码器模块将特殊协议的数据转换成Snort可识别的数据结构，目的是分割数据的包头，利用TCP-IP栈解码并且将包放入一个数据结构中。在抓包和解码工作完成后，将把包送到预处理程序模块来处理，预处理程序对数据包进行检查或者操作后将它们交给下一个模块——检索引擎模块。检索引擎模块对每一个包的某个方面和自己的规则库中的规则进行比对以检测入侵。最后一个组件是输出模块，它对可疑行为产生报警并输出到日志文件中。

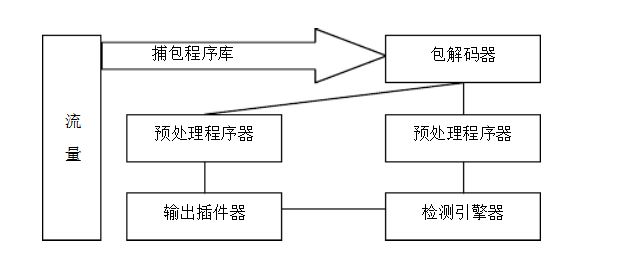


图2 Snort组件数据流程图

**2.4 FP-growth频繁模式挖掘方法**

**2.4.1 关联分析频繁模式挖掘概述**

关联分析用于揭示数据之间的相互关系，而这种关系没有在数据中直接表现出来。关联分析的任务就是发现事物间的关联规则或称相关程度。利用关联分析能寻找数据的相关联系，常用的两种技术为：

关联规则，用于发现一个事物与其他事物间的相互关联性或相互依赖性，如分析客户在超市买牙刷的同时又买牙膏的可能性。

序列模式分析，将重点放在分析数据之间的前后因果关系，如买了电脑的顾客会在三个月内买杀毒软件。

而使用最为活跃和使用最广的研究方法为关联规则。关联分析的一般形式是：如果A发生则B有百分之C在可能发生。C称为关联规则的置信度（Confidence）。关联规则是研究不同类型有物品相互之间关联关系的规则，它最早是针对沃尔玛超市的购物数据分析诞生的，可以用来指导超市进行购销安适。之后应用于其它领域，例如医学病例的共同特征挖掘以及网络入侵检测等等，都可以使用关联规则进行处理。

关联规则最初提出的动机是针对购物篮分析（Market Basket Analysis）问题提出的。假设分店分店经理想要深入了解顾客的购物习惯。特别是想知道有哪些商品，顾客可能会在一次购物时同时购买？为回答该问题，可以对商店的顾客购物零售数量进行购物篮分析。该分析可以通过发现顾客放入“购物篮”的不同商品之间的关联，分析顾客的购物习惯。这种关联的发现可以零售商了解顾客同时频繁购买的商品有哪些，从而零售商开发更好的营销策略。

定义两个相互独立集合X和Y，假设X和Y之间有一定的关联性，即相互之间存在关联规则。而关联规则的表示使用支持度和置信度来说明。

支持度表示X和Y中的项在同一情况下出现的次数。支持度(Support)的公式是：



支持度提示了A与B同时出现的概率。如果A与B同时出现的概率小，说明A与B的关系不大；如果A与B同时出现非常频繁，则说明A与B总是相关的。

置信度表示X和Y在一定条件下出现的概率。置信度（Confidence）的公式是：



置信度提示了A出现时，B是否也会出现或有多大的概率出现。如果置信度为100%，则A和B可以捆绑销售了。如果置信度太低，则说明A的出现与B还是出现关系不大。

关联分析频繁模式挖掘主要有Apriori算法、FP-Growth算法以及Eclat算法。

Apriori是非常经典的关联分析频繁模式挖掘算法，其思想简明，实现方便，只是效率很低，可以作为频繁模式挖掘的入门算法。其主要特点是：

k-1项集连接规律：若有两个k-1项集，每个项集保证有序，如果两个k-1项集的前k-2个项相同，而最后一个项不同，则证明它们是可连接的，可连接生成k项集。

反单调性。如果一个项集是频繁的，那么它的所有子集都是频繁的。即若一个项集的子集不是频繁项集，则该项集肯定也不是频繁项集。

主要算法流程如下：

（1）扫描数据库，生成候选1项集和频繁1项集。

（2）从2项集开始循环，由频繁k-1项集生成频繁k项集。

（3）频繁k-1项集两两组合，判定是否可以连接，若能则连接生成k项集。

（4）对k项集中的每个项集检测其子集是否频繁，舍弃掉子集不是频繁项集即不在频繁k-1项集中的项集。

（5）扫描数据库，计算这一步中过滤后的k项集的支持度，舍弃掉支持度小于阈值的项集，生成频繁k项集。

（6）若当前k项集中只有一个项集时循环结束。

与fp-growth 和apriori算法不同，Eclat算法加入了倒排的思想，具体就是将事务数据中的项作为key，每个项对应的事务ID作为value，加快频繁集生成速度，其算法思想是 由频繁k项集求交集，生成候选k+1项集 。对候选k+1项集做裁剪，生成频繁k+1项集，再求交集生成候选k+2项集。如此迭代，直到项集归一。

Eclat算法过程如下：

（1）一次扫描数据库，获得初始数据。包括频繁1项集，数据库包含的所有items，事务总数（行）transNum，最小支持度minsup=limitValue\*trans。

（2）二次扫描数据库，获得频繁2项集。

（3）按照Eclat算法，对频繁2项集迭代求交集，做裁剪，直到项集归一。

Apriori算法是关联算法中比较经典在算法。它便于理解和程序代码实现，因此在一般数据处理和数据的挖掘中应用非常广泛，但是由于它在算法设计上具有很大的局限性，并不能较为合适地处理大数据。

最主要的是Apriori使用的是Apriori这一特性来生成频繁项候选集，这样做的好处是在单机在情况下，可以对频繁项集进行压缩处理，从而在有限的内存情况下最大限度地提高了去处效率。但是这样做带来好处同时还存在着两个主要问题：

第一个问题就是会产生较多的小频繁项，小频繁项集过多使得数据在进行计算处理的时候效率极大地降低，从而使得复杂度以指数形式增长，降低了Apriori算法全体效率。

第二个问题就是由于频繁项集处理需要多次扫描原样本数据库，而一般情况下IO的处理需要消耗大量的处理时间，从而算法在计算的过程中消耗大量的资源在数据的读取上。

基于以上Apriori算法的不足，一个新的关联算法被提出来，即FT树算法（FP-Growth)。

**2.4.2 FP-growth频繁模式挖掘基本原理**

众所周知，[Apriori](http://baike.baidu.com/subview/2230129/2230129.htm)算法在产生频繁模式完全集前需要对数据库进行多次扫描，同时产生大量的候选频繁集[23][24]，这就使Apriori算法时间和[空间复杂度](http://baike.baidu.com/subview/540497/540497.htm)较大。但是Apriori算法中有一个很重要的性质：频繁项集的所有[非空子集](http://baike.baidu.com/subview/2789635/2789635.htm)都必须也是频繁的。但是Apriori算法在挖掘额长频繁模式的时候性能往往低下，Jiawei Han提出了FP-Growth算法。

基本思路：不断地[迭代](http://baike.baidu.com/subview/461623/461623.htm)FP-tree 的构造和投影过程，FP-growth算法试图解决多次扫描数据库从而带来的大量小频繁项集问题。这个算法的理论上只对数据库进行再次扫描，直接压缩数据库生成一个频繁模式树从而形成规则。

FP-Growth算法主要步骤如下:

（1）扫描数据库DB一遍.得到频繁项的集合F和每个频繁项的支持度.把F按支持度递降排序,结果记为L。

（2）创建FP-tree的根节点,记为T,并且标记为’null’.然后对DB中的每个事务Trans做如下的步骤.

（3）根据L中的顺序,选出并排序Trans中的事务项.把Trans中排好序的事务项列表记为[p|P],其中p是第一个元素,P是列表的剩余部分.调用insert\_tree([p|P],T)。

函数insert\_tree([p|P],T)的运行如下：如果T有一个子结点N,其中N.item-name=p.item-name,则将N的count域值增加1；否则,创建一个新节点N,使它的count为1,使它的父节点为T,并且使它的node\_link和那些具有相同item\_name域串起来。如果P非空,则递归调用insert\_tree(P,N)。

**2.5 本章小结**

本章主要介绍了Spark框架结构和运行原理，讨论了聚类分析算法以及K-means算法，给出了入侵检测系统的组成和基本原理，探讨了关联分析算法以及FP-growth算法，其中重点介绍了数据挖掘中的Spark框架、聚类分析中的K-means算法、关联分析中FP-growth算法和入侵检测系统中的Snort入侵检测系统。

# 第三章 Spark框架下基于k-means的流量异常检测研究与实现

**3.1 Spark框架下基于k-means的流量异常检测基本思路**

**3.1.1 大数据处理平台选择基本思路**

机器学习旨在使计算机具有人类一样的学习能力和模仿能力，这也是实现人工智能的核心思想和方法。传统的机器学习算法，由于技术和单机存储的限制，只能在少量数据上使用，随着 HDFS(Hadoop Distributed File System) 等分布式文件系统出现，存储海量数据已经成为可能。

目前过程开发中常用的有两种大数据平台，一个是Hadoop,另一个就是Spark。Hadoop，分布式批处理计算，强调批处理，常用于数据挖掘、分析等方向的研究和使用。而Spark是基于内存计算的开源的集群计算系统，目的是让数据分析更加快速。Spark是一种与Hadoop相似的开源集群计算环境，但是两者之间还存在一些不同之处，这些有用的不同之处使Spark在某些工作负载方面表现得更加优越，换句话说，Spark启用了内存分布数据集，除了能够提供交互式查询外，它还可以优化迭代工作负载。

Hadoop由于 MapReduce 算法自身的限制，使得使用 MapReduce 来实现分布式机器学习算法非常耗时和消耗磁盘容量。因为通常情况下机器学习算法参数学习的过程都是迭代计算的，即本次计算的结果要作为下一次迭代的输入， 这个过程中，如果使用 MapReduce，我们只能把中间结果存储磁盘，然后在下一次计算的时候从新读取，这对于迭代频发的算法显然是致命的性能瓶颈

Spark是在Scala语言中实现的，它将Scala用作其应用程序框架。与Hadoop不同，Spark和Scala能够紧密集成，其中的Scala可以像操作本地集合对象一样轻松地操作分布式数据集。

尽管创建Spark是为了支持分布式数据集上的迭代作业，但是实际上它是对Hadoop的补充，可以在Hadoop文件系统中并行运行。通过名为 Mesos的第三方集群框架可以支持此行为。Spark由加州大学伯克利分校AMP实验室 (Algorithms,Machines,andPeopleLab)开发，可用来构建大型的、低延迟的数据分析应用程序。

虽然Spark与Hadoop有相似之处，但它提供了具有有用差异的一个新的集群计算框架。首先，Spark是为集群计算中的特定类型的工作负载而设计，即那些在并行操作之间重用工作数据集（比如机器学习算法）的工作负载。为了优化这些类型的工作负载，Spark引进了内存集群计算的概念，可在内存集群计算中将数据集缓存在内存中，以缩短访问延迟.

在大数据处理方面相信大家对Hadoop已经耳熟能详，基于GoogleMap/Reduce来实现的Hadoop为开发者提供了map、reduce原语，使并行批处理程序变得非常地简单和优美。Spark提供的数据集操作类型有很多种，不像Hadoop只提供Map和Reduce两种操作。如 map,filter,flatMap,sample,groupByKey,reduceByKey,union,join,cogroup,mapValues,sort,partionBy 等多种操作类型，他们把这些操作称为Transformations。同时还提供Count,collect,reduce,lookup,save等多种actions。这些多种多样的数据集操作类型，给上层应用者提供了方便。各个处理节点之间的通信模型不再像Hadoop那样就是唯一的 DataShuffle一种模式。用户可以命名，物化，控制中间结果的分区等。

所以可以说Spark编程模型比Hadoop更灵活。

Spark配有一个流数据处理模型，与Twitter的Storm框架相比，Spark采用了一种有趣而且独特的办法。Storm基本上是像是放入独立事务的管道，在其中事务会得到分布式的处理。相反，Spark采用一个模型收集事务，然后在短时间内（我们假设是5秒）以批处理的方式处理事件。所收集的数 据成为他们自己的RDD，然后使用Spark应用程序中常用的一组进行处理。作者声称这种模式是在缓慢节点和故障情况下会更加稳健，而且5秒的时间间隔通常对于大多数应用已经足够快了。这种方法也很好地统一了流式处理与非流式处理部分。

**3.1.2 聚类方法选择基本思路**

聚类分析是一种静态数据分析方法，常被用于机器学习，模式识别，数据挖掘等领域。通常认为，聚类是一种[无监督](http://zh.wikipedia.org/wiki/%E9%9D%9E%E7%9B%A3%E7%9D%A3%E5%BC%8F%E5%AD%B8%E7%BF%92)式的机器学习方法，它的过程是这样的：在未知样本类别的情况下，通过计算样本彼此间的距离（欧式距离,马式距离，汉明距离，余弦距离等）来估计样本所属类别。从结构性来划分，聚类方法分为自上而下和自下而上两种方法，前者的算法是先把所有样本视为一类，然后不断从这个大类中分离出小类，直到不能再分为止；后者则相反，首先所有样本自成一类，然后不断两两合并，直到最终形成几个大类：

Connectivity based clustering

Centroid-based clustering

Distribution-based clustering

Density-based clustering

K-Means就属于第二类聚类算法。K-means聚类是一种自下而上的聚类方法，它的优点是简单、速度快；缺点是聚类结果与初始中心的选择有关系，且必须提供聚类的数目。K-means的第二个缺点是致命的，因为在有些时候，我们不知道样本集将要聚成多少个类别，这种时候K-means是不适合的，推荐使用hierarchical 或meanshift来聚类。第一个缺点可以通过多次聚类取最佳结果来解决。

就因为K-Means算法的快速性和简单性，在对大数据的处理过程中可以更加快速和减少计算量。于是选用K-Means算法。

**3.2 Spark框架下基于k-means的流量异常检测方法**

K-Means的目标是一组数据点划分成K个簇。首先，设置初始K个聚类中心，我们可以发现哪个集群和每个数据点是最接近的。然后，我们计算每个新簇的平均值，并使用结果来更新我们的聚类中心。然后再重新分配和更新，我们每个集群中的点互相越来越相似与彼此。通过重复这两个步骤之间的迭代，我们通常可以收敛到一个很好的解决方案。

在Spark函数库MLlib中的K-Means算法具有以下参数：

class KMeans private (

private var k: Int,

private var maxIterations: Int,

Private var runs: Int,

Private var initializationMode: String

Private var iniaializationSteps: Int

Private var epsilon: Double,

Private var seed:Log)

extends Serializable with Logging

其中，k 表示期望的聚类的个数，maxInterations 表示方法单次运行最大的迭代次数，runs 表示算法被运行的次数，K-means算法不保证能返回全局最优的聚类结果，所以在目标数据集上多次跑K-means算法，有助于返回最佳聚类结果，initializationMode表示初始聚类中心点的选择方式, 目前支持随机选择或者 K-means||方式，默认是K-means||，initializationSteps表示 K-means||方法中的部数，epsilon表示K-means算法迭代收敛的阀值，seed表示集群初始化时的随机种子。

通常，先调用KMeans.train方法对数据集进行训练，这个方法会返回KMeansModel类实例，然后我们可以使用 KMeansModel.predict方法对新的数据点进行所属聚类的预测。KMeansModel.predict 方法接受不同的参数，可以是向量，或者 RDD，返回是入参所属的聚类的索引号。

**3.3 Spark框架下基于k-means的流量异常检测技术实现**

设计目标是实现对访问流量的分析，并预测攻击行为的发生。把处理好的数据放入大数据平台进行并行计算，系统框架如图3所示。



图3 流量异常检测系统框架

如图3所示，整个处理过程主要分为6个部分。

（1）载入数据模块。通过SparkContext将存储在本地的流量文件导入Spark RDD中。方便下一步的处理和分析。

（2）数据预处理模块。这一步主要负责将流量文件中的非数值型数据替换成数值型数据，方便下一步的计算和研究。

（3）数据规整化模块。由于上一步产生的数据存在矩阵过于稀疏的情况，这一步将实现稀疏矩阵转换成密集矩阵。

（4）机器训练模块。使用机器学习技术训练得到模型，并通过计算各点和聚类中心的距离的最小值，得到最优的聚类模型。

（5）预测模块。使用训练好的模型对流量数据进行预测和分类。

（6）比对模块。预测出的每条流量数据所属的类簇，并和标签比对，计算正确性。

**3.4 实验分析**

**3.4.1 实验数据集及性能指标**

这次处理对象为KDD-99数据集，KDD-99数据集是林肯实验室在1999年采集的流量数据。KDD-99的任务是建立一个网络入侵探测器,预测模型能够区分“坏”连接,称为入侵或攻击,“好”的正常连接。这个数据库包含一组标准的审计数据,其中包括各种各样的网络环境模拟的军事入侵。

类簇准确率*p*计算公式为



其中*m*为类簇中数量占第一位的数据的个数，*w*为类簇中所有数据的个数。

总准确率*P*计算公式：



其中为所有类簇中第一位数据的和，为所有类簇中所有的数据的和。

**3.4.2 实验环境及配置**

实验在Linux系统的支系Ubuntu 14.04LTS上完成，配置主要有Spark的安装与测试, Python的安装与测试, Scala的安装与测试, IPython的安装与测试等。

（1）Spark的安装与测试

从Spark官网上下载最新的Spark安装包，使用tar解压到Home目录下，然后进入Spark解压后的目录，打开ternimal输入命令：

./bin/spark-shell

打开Spark的交互式shell，从图中可以看出界面中明显出现了Spark的字母，如图4所示：

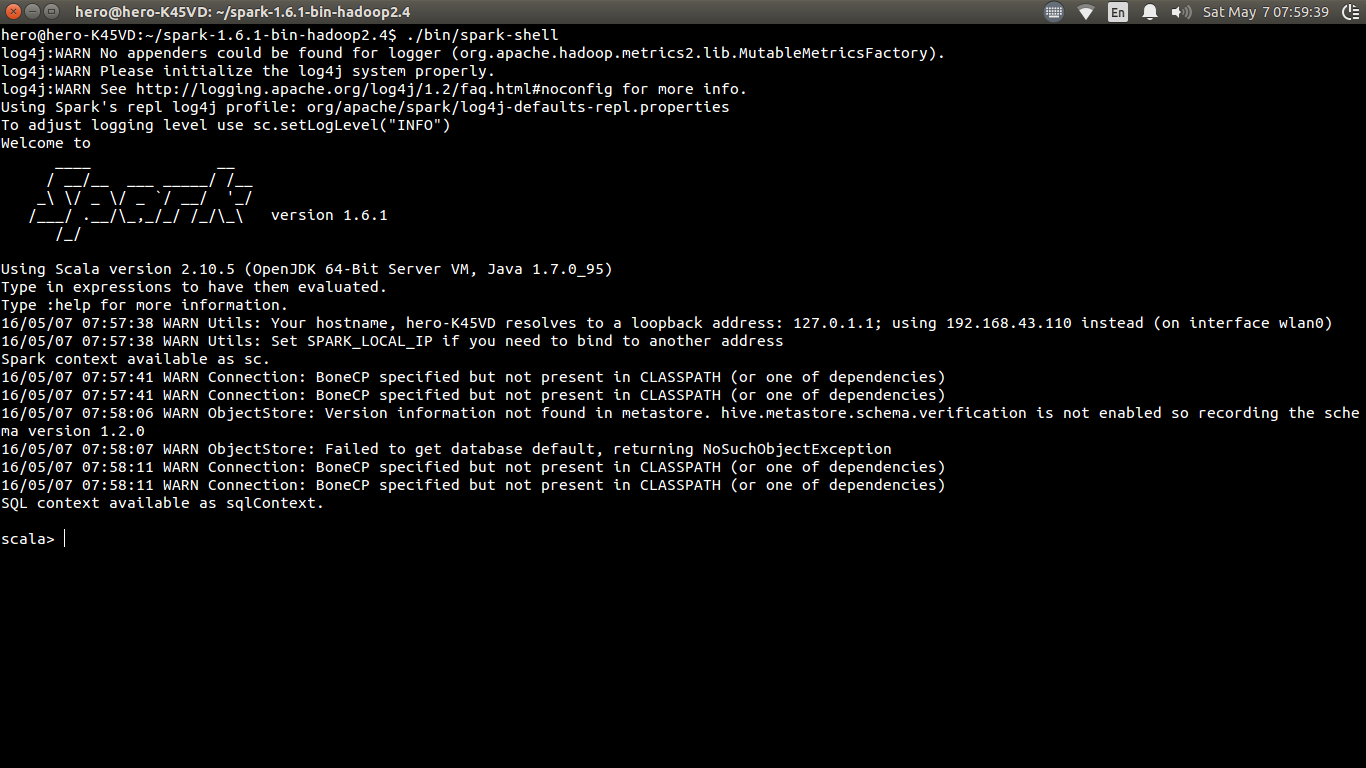


图4

看到Spark字母就说明Spark已经安装好并成功启动了。

下面测试Spark的计算功能。

在terminal中输入：

./bin/run-example org.apache.spark.examples.SparkPi

使用测试文件测试Spark的安装成功与否，Spark成功启动并计算出了π的值为3.1371，结果如下图5：

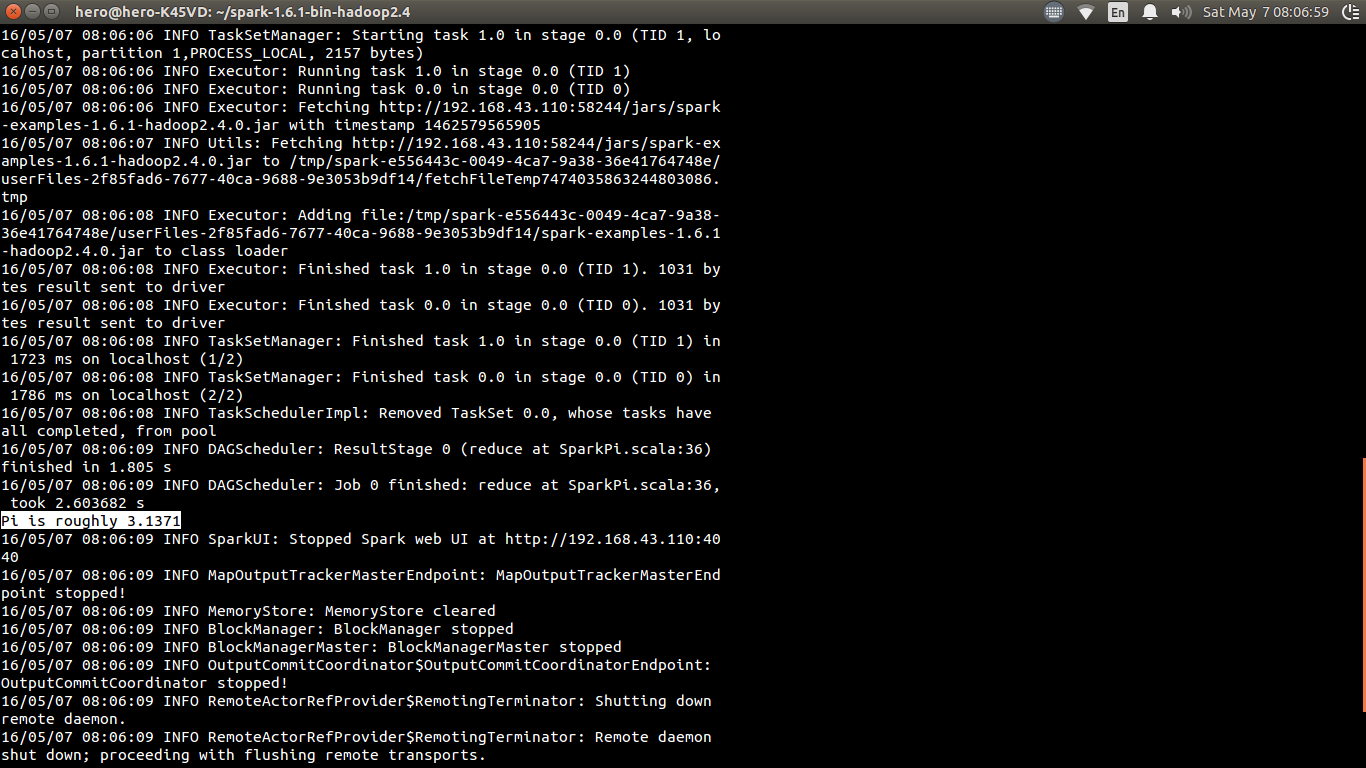


图5

可以看出Spark已经将计算好的结果显示出来，到这里为止，Spark的安装与测试完成。

1. Python的安装与测试

打开terminal，输入：

sudo apt-get install python

apt-get是debian系统自带的软件管理命令，执行这个命令，系统将会自动帮你安装Python。

安装完之后打开terminal，输入：

python

从terminal中进入了Python的交互式界面中，如下图6所示

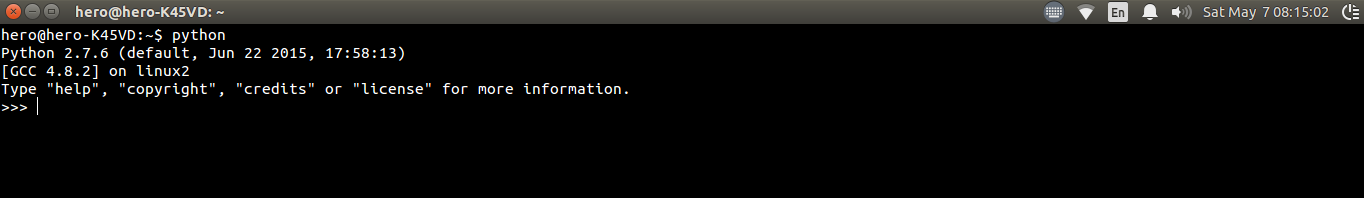


图6

Python安装成功。

测试时，输入：

2+3

然后回车，Python会识别你的输入并帮你计算显示结果数值。

（3）Scala的安装与测试

打开terminal输入：

sudo apt-get install scala

系统开始下载scala。

下载完毕，打开Scala。如下图7所示：

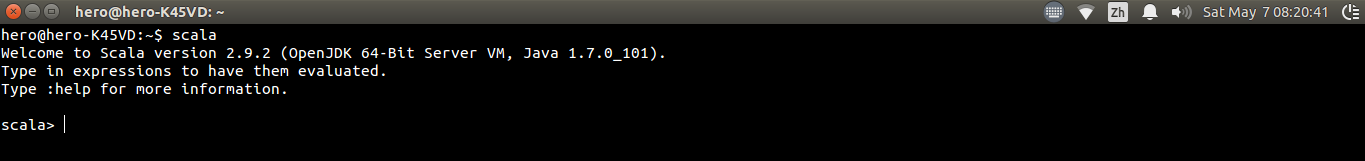


图7

Scala安装成功。

测试依旧输入：

2+3

然后回车，Scala会显示最后的结果为5。

1. IPython的安装与测试

输入：

sudo apt-get install ipython

然后在terminal中输入：

ipython

进入IPython，如图8：

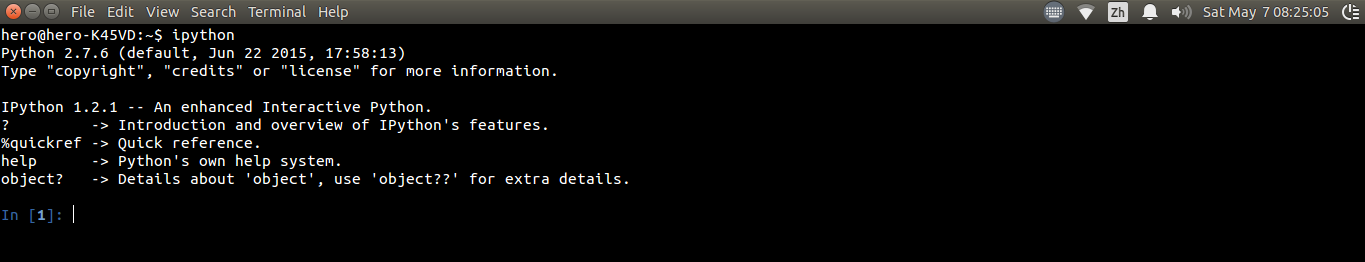


图8

IPython安装成功,测试方法同上。

**3.4.3 实验运行过程**

1. 载入数据和数据的预处理

首先使用的Python和IPython进行数据的载入和预处理。

KDD-99数据集中的原始数据中包含了非数值型的数据，而K-means是没法计算和处理非数值型的数据的，为了使得计算更能体现数据本身所蕴含的信息，所以我们要将其中的非数值型数据悉数替换成数值型的。如果单纯的舍弃这些数据，将使得计算出现很大的误差。

官方的KDD-99原始数据是这样的：

0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,normal.

观察不难发现，非数值型的数据也就第2、3、4和最后一项，我们将除了最后一项的数据都替换成数值。

首先我们在程序中虚拟建立一个对应表，将其中的非数值型数据按出现的顺序一一替换成数字，表中是第2、3、4项的所有可能出现的集合，我们可以统计数据的所有集合后建立，建立完毕后结果如下：

第2项数据的对应表：

[“tcp”, “udp”, “icmp”]

第3项数据的对应表：

[“aol”, “auth”, “bgp”, “courier”, “csnet\_ns”, “ctf”, “daytime”, “discard”, “domain”, “domain\_u”, “echo”, “eco\_i”, “ecr\_i”, “efs”, “exec”, “finger”, “ftp”, “ftp\_data”, “gopher”, “harvest”, “hostnames”, “http”, “http\_2784”, “http\_443”, “http\_8001”, “imap4”, “IRC”, “iso\_tsap”, “klogin”, “kshell”, “ldap”, “link”, “login”, “mtp”, “name”, “netbios\_dgm”, “netbios\_ns”, “netbios\_ssn”, “netstat”, “nnsp”, “nntp”, “ntp\_u”, “other”, “pm\_dump”, “pop\_2”, “pop\_3”, “printer”, “private”, “red\_i”, “remote\_job”, “rje”, “shell”, “smtp”, “sql\_net”, “ssh”, “sunrpc”, “supdup”, “systat”, “telnet”, “tftp\_u”, “tim\_i”, “time”, “urh\_i”, “urp\_i”, “uucp”, “uucp\_path”, “vmnet”, “whois”, “X11”, “Z39\_50”]

第4项数据的对应表：

[“OTH”, “REJ”, “RSTO”, “RSTOS0”, “RSTR”, “S0”, “S1”, “S2”, “S3”, “SF”, “SH”]

然后便是使用程序查找与替换。

替换后的数据如下：

0,1,22,10,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,normal.

不难发现，非数值型数据除了最后一项都不存在了，现在就可以带入Spark中进行聚类运算。

（2）数据规整化与机器训练

数据的规整化，使用Spark自带的数据处理函数StandardScaler()完成，观察数据不难发现，数据本身是比较稀疏的，我们可以通过这个函数将稀疏的数据转换成密集的数据。

然后将转换后的数据带入聚类算法函数KMeans中训练，并输出结果。

这里使用了一个简单的函数，通过迭代计算某个范围内的K值，并计算这些K值下的各簇每个点到各自的聚类中心的聚类的平均值，推断最佳的K值。

计算完毕，输出保存结果。

**3.4.4 结果对比分析**

1. 设定最大K值为30并将非数值型数据替换成数值后的结果。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类簇编号 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 攻击类型 | normal | neptune | smurf | neptune | normal | smurf | smurf |
| 代表攻击数 | 5340 | 2252 | 548 | 47221 | 8109 | 8691 | 10411 |
| 非代表攻击数 | 423 | 4 | 0 | 0 | 354 | 102 | 3 |
| 总数 | 5763 | 2256 | 548 | 47221 | 8463 | 8793 | 10414 |
| 准确率 | 92.66% | 99.82% | 100.00% | 100.00% | 95.82% | 98.84% | 99.97% |
| 类簇编号 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 攻击类型 | teardrop | normal | normal | normal | ipsweep | neptune | smurf |
| 代表攻击数 | 970 | 33184 | 11536 | 5258 | 1135 | 37266 | 34533 |
| 非代表攻击数 | 0 | 1098 | 26 | 63 | 327 | 36 | 1 |
| 总数 | 970 | 34282 | 11562 | 5321 | 1462 | 37302 | 34534 |
| 准确率 | 100.00% | 96.80% | 99.78% | 98.82% | 77.63% | 99.90% | 100.00% |
| 类簇编号 | 14 | 15 | 16 | 17 | 18 |
| 攻击类型 | normal | normal | neptune | normal | smurf |
| 代表攻击数 | 29015 | 4039 | 20457 | 424 | 193199 |
| 非代表攻击数 | 1640 | 390 | 2267 | 322 | 0 |
| 总数 | 30655 | 4429 | 22724 | 746 | 193199 |
| 准确率 | 94.65% | 91.19% | 90.02% | 56.84% | 100.00% |

表1 K=30维度=41

从表1中可以看出，聚类分析后自动将数据分成[0-18]的19簇。

第0簇主要由normal组成，准确率p0为91.66%；第1簇主要由neptune组成，准确率p1为99.82%；第2簇主要由smurf组成，准确率p2为100%；第3簇主要由neptune组成，准确率p3为100%；第4簇主要由normal组成，准确率p4为95.82%。

计算总准确率，为98.47%。

从表中也可以看出，使用的算法来进行K值选取的优化，对于提高聚类效果是可行的。

1. 设定最大K值为60并将非数值型数据替换成数值后的结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类簇编号 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 攻击类型 | smurf | smurf | satan | normal | neptune | smurf | smurf |
| 代表攻击数 | 70 | 548 | 1400 | 4026 | 2182 | 54 | 33368 |
| 非代表攻击数 | 51 | 0 | 174 | 183 | 3 | 5 | 0 |
| 总数 | 121 | 548 | 1574 | 4209 | 2185 | 59 | 33368 |
| 准确率 | 57.85% | 100.00% | 88.95% | 95.65% | 99.86% | 91.53% | 100.00% |
| 类簇编号 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 攻击类型 | null | smurf | normal | neptune | smurf | neptune | teardrop |
| 代表攻击数 | 0 | 33 | 7739 | 10097 | 1154 | 11672 | 970 |
| 非代表攻击数 | 0 | 0 | 111 | 46 | 0 | 20 | 0 |
| 总数 | 0 | 33 | 7850 | 10143 | 1154 | 11692 | 970 |
| 准确率 | 0.00% | 100.00% | 98.59% | 99.55% | 100.00% | 99.83% | 100.00% |
| 类簇编号 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 攻击类型 | normal | neptune | normal | null | null | ipsweep | null |
| 代表攻击数 | 1926 | 10360 | 26607 | 0 | 0 | 1135 | 0 |
| 非代表攻击数 | 178 | 178 | 623 | 0 | 0 | 190 | 0 |
| 总数 | 2104 | 10538 | 27230 | 0 | 0 | 1325 | 0 |
| 准确率 | 91.54% | 98.31% | 97.71% | 0.00% | 0.00% | 85.66% | 0.00% |
| 类簇编号 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 攻击类型 | normal | smurf | null | smurf | neptune | portsweep | smurf |
| 代表攻击数 | 25970 | 4073 | 0 | 44 | 14211 | 636 | 6662 |
| 非代表攻击数 | 1550 | 0 | 0 | 0 | 0 | 20 | 0 |
| 总数 | 27520 | 4073 | 0 | 44 | 14211 | 656 | 6662 |
| 准确率 | 94.37% | 100.00% | 0.00% | 100.00% | 100.00% | 96.95% | 100.00% |
| 类簇编号 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| 攻击类型 | smurf | normal | neptune | smurf | neptune | normal | null |
| 代表攻击数 | 9 | 24 | 9379 | 3688 | 24773 | 824 | 0 |
| 非代表攻击数 | 0 | 30 | 0 | 0 | 6 | 75 | 0 |
| 总数 | 9 | 54 | 9379 | 3688 | 24779 | 899 | 0 |
| 准确率 | 100.00% | 44.44% | 100.00% | 100.00% | 99.98% | 91.66% | 0.00% |
| 类簇编号 | 35 | 36 | 37 | 38 | 39 | 40 | 41 |
| 攻击类型 | null | smurf | neptune | smurf | neptune | normal | null |
| 代表攻击数 | 0 | 193081 | 12927 | 560 | 11473 | 371 | 0 |
| 非代表攻击数 | 0 | 0 | 0 | 6 | 0 | 314 | 0 |
| 总数 | 0 | 193081 | 12927 | 566 | 11473 | 685 | 0 |
| 准确率 | 0.00% | 100.00% | 100.00% | 98.94% | 100.00% | 54.16% | 0.00% |
| 类簇编号 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 攻击类型 | null | normal | portsweep | smurf | normal | normal | null |
| 代表攻击数 | 0 | 2826 | 1 | 387 | 7546 | 5241 | 0 |
| 非代表攻击数 | 0 | 286 | 0 | 68 | 310 | 11 | 0 |
| 总数 | 0 | 3112 | 1 | 455 | 7856 | 5252 | 0 |
| 准确率 | 0.00% | 90.81% | 100.00% | 85.05% | 96.05% | 99.79% | 0.00% |
| 类簇编号 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 攻击类型 | smurf | null | normal | null | normal | smurf | guess\_passwd |
| 代表攻击数 | 2522 | 0 | 9627 | 0 | 4141 | 26429 | 52 |
| 非代表攻击数 | 0 | 0 | 507 | 0 | 15 | 0 | 12 |
| 总数 | 2522 | 0 | 10134 | 0 | 4156 | 26429 | 64 |
| 准确率 | 100.00% | 0.00% | 95.00% | 0.00% | 99.64% | 100.00% | 81.25% |
| 类簇编号 | 56 | 57 | 58 | 59 |
| 攻击类型 | null | null | null | smurf |
| 代表攻击数 | 0 | 0 | 0 | 8089 |
| 非代表攻击数 | 0 | 0 | 0 | 0 |
| 总数 | 0 | 0 | 0 | 8089 |
| 准确率 | 0.00% | 0.00% | 0.00% | 100.00% |

表2 K=60维度=41

最后可以算出的值为98.99%。

从与的对比可以看出，通过扩大最大K值的取值范围和平均距离对K值的筛选，可以有效提高准确率一个百分点。

1. 设定最大K值为30并没有对非数值型数据进行处理的结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类簇编号 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 攻击类型 | normal | normal | satan | guess\_passwd | normal | smurf | smurf |
| 代表攻击数 | 3872 | 7010 | 1398 | 52 | 1062 | 11324 | 7700 |
| 非代表攻击数 | 189 | 215 | 196 | 77 | 76 | 37 | 3 |
| 总数 | 4061 | 7225 | 1594 | 129 | 1138 | 11361 | 7703 |
| 准确率 | 95.35% | 97.02% | 87.70% | 40.31% | 93.32% | 99.67% | 99.96% |
| 类簇编号 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 攻击类型 | normal | null | neptune | normal | ipsweep | neptune | smurf |
| 代表攻击数 | 31773 | 0 | 37501 | 2677 | 1135 | 23743 | 34478 |
| 非代表攻击数 | 1035 | 0 | 25 | 204 | 369 | 0 | 1 |
| 总数 | 32808 | 0 | 37526 | 2881 | 1504 | 23743 | 34479 |
| 准确率 | 96.85% | 0.00% | 99.93% | 92.92% | 75.47% | 100.00% | 100.00% |
| 类簇编号 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 攻击类型 | smurf | smurf | normal | smurf | smurf | normal | normal |
| 代表攻击数 | 809 | 81 | 3938 | 548 | 46 | 1850 | 26481 |
| 非代表攻击数 | 313 | 9 | 394 | 0 | 1 | 199 | 1373 |
| 总数 | 1122 | 90 | 4332 | 548 | 47 | 2049 | 27854 |
| 准确率 | 72.10% | 90.00% | 90.90% | 100.00% | 97.87% | 90.29% | 95.07% |
| 类簇编号 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 攻击类型 | neptune | neptune | smurf | teardrop | smurf | smurf | normal |
| 代表攻击数 | 25351 | 20457 | 24 | 970 | 33368 | 5 | 1825 |
| 非代表攻击数 | 3 | 5914 | 0 | 0 | 0 | 0 | 165 |
| 总数 | 25354 | 26371 | 24 | 970 | 33368 | 5 | 1990 |
| 准确率 | 99.99% | 77.57% | 100.00% | 100.00% | 100.00% | 100.00% | 91.71% |
| 类簇编号 | 28 |
| 攻击类型 | normal |
| 代表攻击数 | 10537 |
| 非代表攻击数 | 126 |
| 总数 | 10663 |
| 准确率 | 98.82% |

表3 K=30维度=38

的值算出为97.79%。

最后通过比较的98.47%与的97.79%可以看出将非数值型数据转换成数值型数据后使得计算维度的增加可以有效提高K-means的预测攻击类型的准确率。

**3.5 本章小结**

对于K值选取的问题，一直是k-means算法的难题，本实验通过迭代计算不同K值中各点到计算出的中心点之间的距离的平均值，推测K值的范围，并比较了不同K最大值的情况下，不同的数据维度下，对数据结果的影响。

# 第4章 Spark框架下基于FP-growth的告警关联研究与实现

**4.1 Spark框架下基于FP-growth的告警事件关联基本思路**

传统的Aprioti算法存在以下瓶颈：对数据库的扫描次数过多。当事务数据库中存放大量事务数据时，在有限的内存容量下，系统I/O负载相当大。对每次k循环，候选集CK中的每个元素都必须通过扫描数据库一次来验证其是否加入LK。假如有一个频繁大项集包含10个项的话，那么就至少需要扫描事务数据库10遍。每次扫描数据库的时间就会非常长，这样导致Apriori算法效率相对低。

可致使庞大侯选集的产生。由LK-1产生k-侯选集CK是指数增长的，例如104的1-频繁项集就有可能产生接近107个元素的2-侯选集。如果要产生一个很长的规则时，产生的中间元素也是巨大的。

基于支持度和可信度框架理论发现的大量规则中，有一些规则即使满足用户指定的最小支持度和可信度，但仍没有实际意义；如果最小支持度阈值定得越高，有用数据就越少，有意义的规则也就不易被发现，这样会影响决策的制定。

算法适应范围小。Apriori算法仅仅考虑了布尔型的单维关联规则的挖掘，在实际应用中，可能出现多类型的、多维的、多层的关联规则。

**4.2 Spark框架下基于FP-growth的告警事件关联方法**

针对Apriori算法的瓶颈问题，2000年Han等人提出了一个称为FP-growth的算法这个算法只进行2次数据库扫描。它不使用侯选集，直接压缩数据库成一个频繁模式树，最后通过这棵树生成关联规则。研究表明它比Apriori算法快一个数量级。FP-growth算法是一种不产生候选模式而采用频繁模式增长的方法挖掘频繁模式的算法。它首先通过扫描事务集产生一个紧凑的、包含D中频繁模式全部信息的频繁模式树（Frequent Pattern tree），然后通过对FP-tree的挖掘获得频繁模式。

FP-Growth的算法步骤：

FP-tree构建：

（1）第一遍扫描数据，找出频繁1项集L，按降序排序

（2）第二遍扫描数据：

对每个transaction，过滤不频繁集合，剩下的频繁项集按L顺序排序。

把每个transaction的频繁1项集插入到FP-tree中，相同前缀的路径可以共用，同时增加一个header table，把FP-tree中相同item连接起来，也是降序排序。

频繁项挖掘的过程如下：

（1）从header table的最下面的item开始，构造每个item的条件模式基（conditional pattern base）。

（2）顺着header table中item的链表，找出所有包含该item的前缀路径，这些前缀路径就是该item的条件模式基（CPB）。

（3）所有这些CPB的频繁度（计数）为该路径上item的频繁度（计数）。

（4）如包含p的其中一条路径是fcamp，该路径中p的频繁度为2，则该CPB fcam的频繁度为2。

构造条件FP-tree（conditional FP-tree）的过程如下：

（1）累加每个CPB上的item的频繁度（计数），过滤低于阈值的item，构建FP-tree。

（2）如m的CPB{<fca:2>, <fcab:1>}，f:3, c:3, a:3, b:1, 阈值假设为3，过滤掉b。

FP-Growth的过程如下：

递归的挖掘每个条件FP-tree，累加后缀频繁项集，直到找到FP-tree为空或者FP-tree只有一条路径（只有一条路径情况下，所有路径上item的组合都是频繁项集）。

**4.3 实验分析**

**4.3.1 实验数据集及性能指标**

（1）DARPA 2000数据集概述

MIT林肯实验室下的The Cyber Systems and Technology Group研究组（前身为DARPA入侵检测评估组），是由美国国防高级研究项目机构（DARPA ITO）和美国空军研究实验室（AFRL / SNHS）赞助，其主要工作是收集和分布式计算机网络的入侵检测系统评估的标准库，并且协同空军研究实验室，完成了世界上第一个正式的，可重复的，和统计上的显着的入侵检测系统的评价。这样的评价工作已经进行了1998和1999年两年。

测试中每个系统的检测概率和虚警概率都是控制得很小心的，这些结论提供了显着的入侵检测研究领域的研究工作提供方向和客观的技术状态的标准。

1. LLDOS 1.0概述

这个攻击场景数据集是创建作为DARPA数据集的一部分。它包括一个分布式拒绝服务攻击，由一个新手攻击运行。这个版本的未来版本将包含更多的隐蔽攻击。

这种攻击场景是在多个网络和审计会话中进行的。这些课程被分为5个攻击阶段，在这一过程中，攻击者对网络进行了一系列探测，利用Solaris sadmind漏洞，安装木马Mstream DDoS软件，并发起DDoS攻击在主机网站的服务器。

攻击的前提是，相对其他黑客为新手的对手试图展现他或者她利用脚本攻击闯入各种互联网主机的能力，安装的组件并运行一个分布式拒绝服务攻击平台，然后发动DDoS攻击在美国政府网站。攻击者使用的是Solaris sadmind漏洞，一个众所周知的远程root漏洞，攻击成功并获取了三台在Eyrie 空军基地的Solaris主机的root访问权限。这些攻击成功是由于美国空军基地应用的是较差的安全模型，安全模型中包括许多服务，包括危险的代理通过防火墙访问内网的“SUNRPC”服务。攻击者使用Mstream DDoS工具，一个不太复杂的DDoS攻击工具。它不使用加密和不提供广泛的攻击选择其他工具，如TribeFloodNetwork或Trinoo。一个Mstream “server”软件，通过安装在僵尸服务器上用于生成和发送DDoS攻击数据包的，现在Mstream “server”已经安装在了三个受害主机上，而Mstream“master”软件提供了一个用户界面和控制安装在受害者上的Mstream “server”。

攻击场景的五个阶段是：

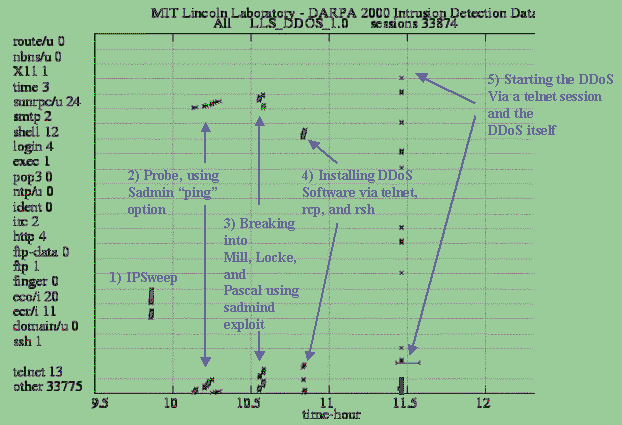
从远程站点对美国空军基地进行Ping扫射（ipsweep）

彻底调查所有存活的IP并找寻运行了sadmind的Solaris主机

通过sadmind漏洞进入主机都有成功和不成功的

安装木马Mstream DDoS软件在美国空军基地的三台主机上

发动DDoS攻击

 图9 DDOS攻击时间表

上面图9的场景，说明了攻击的具体步骤。只有攻击会话被绘制在这里，每一个阶段是记在红色。X轴是东部标准时间时间，Y轴是TCP或UDP服务。哈希标记显示在该时间和该服务的网络会话中的存在的位置。

**4.3.2 实验环境及配置**

本次实验采用Linux debian8.4系统，编译器为gcc和g++。

Linux debian8.4系统安装

从debian官网下载镜像文件，烧制成启动光盘，然后进行安装。

（1）Snort的安装

从Snort官网上下载Snort的LInux安装包，tar解压后进入目录中，然后编译安装：

./configure && make && make install

然后测试Snort，输入：

snort

运行界面将显示一个Snort的图标，如图9所示:



图9

到这一步，Snort安装成功

**4.3.2 实验运行过程**

要实现关联分析，首先得明白关联什么，分析什么，答案是关联网络攻击事件，分析告警日志。

1. 使用Snort和DARPA 2000数据集产生告警日志

第一步便是产生告警日志。

DARPA 2000是美国国防部的2000年对其内网主机模拟攻击尝试的流量数据，格式为tcpdump。

首先我们从林肯实验室的官方网站上下载LLS\_DOS\_1.0\_inside.dump文件，大小为122.1M。

将其放入/etc/snort文件夹中，并开始配置Snort为最严格模式，打开/etc/snort文件夹下的snort.conf文件，使能规则表，使能匹配引擎，使能输出。

在terminal中切换到/etc/snort目录下，输入：

snort -c snort.conf -r LLS\_DDOS\_1.0-inside.dump -A fast

运行过程如下图10所示：

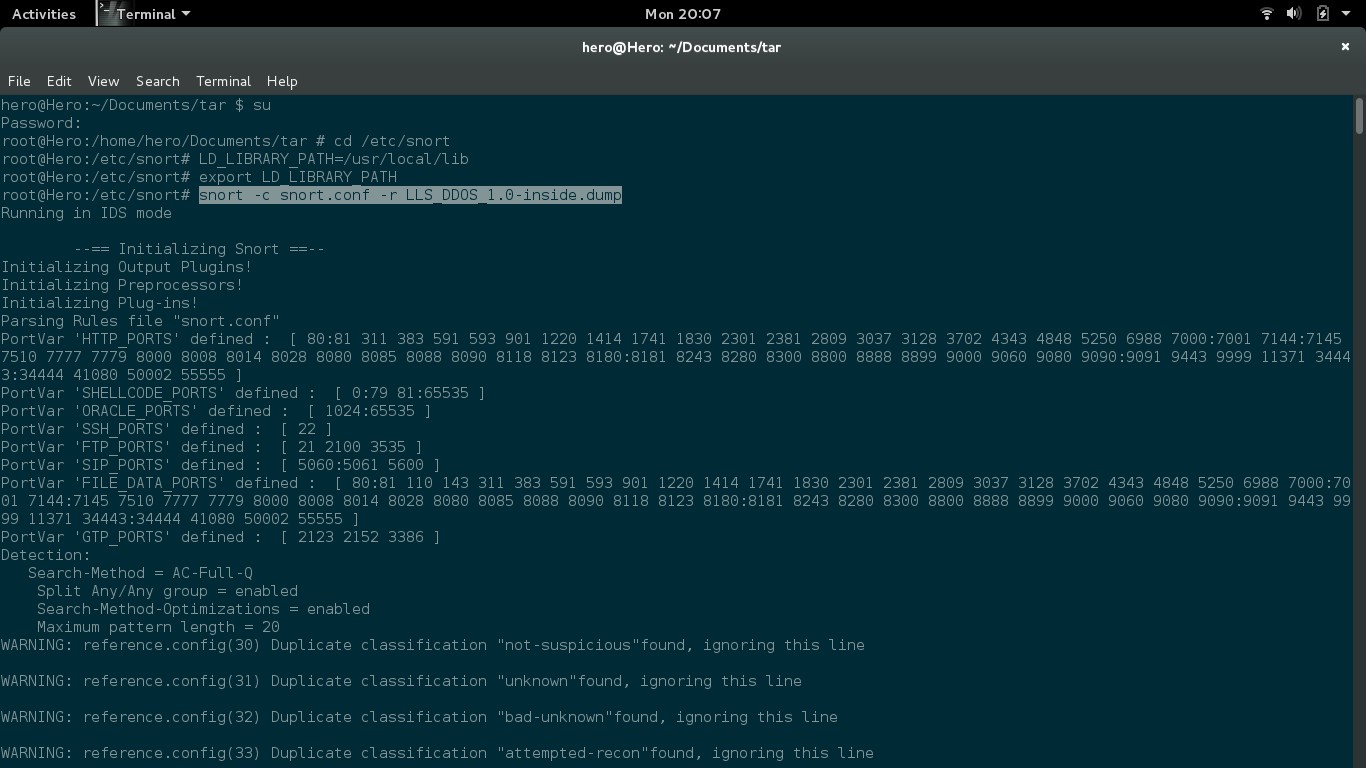


图10

Snort自动使用配置文件Snort.conf进行配置，然后开始进行Snort的初始化，如图11所示：

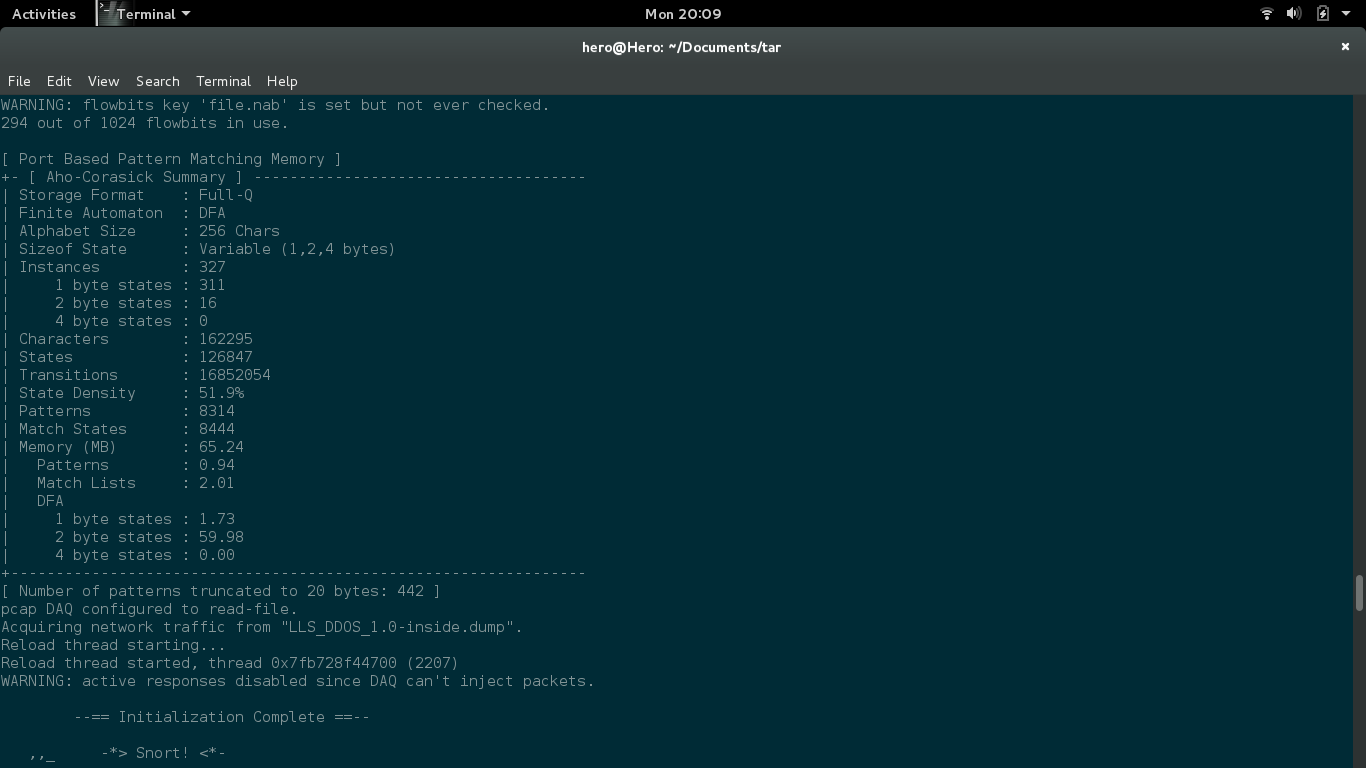


图11

Snort初始化成功，如下图12所示：

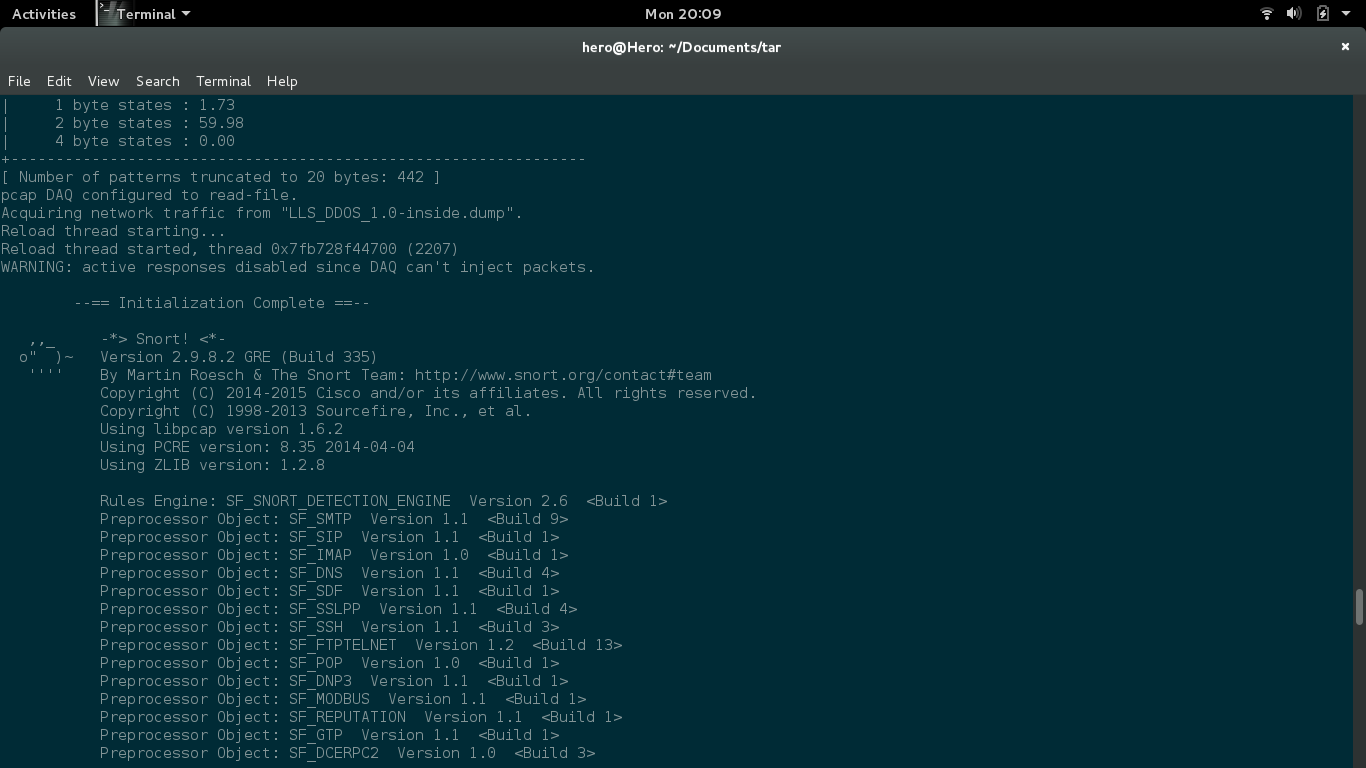


图12

分析dump流量文件中各成分组成如下图13所示：

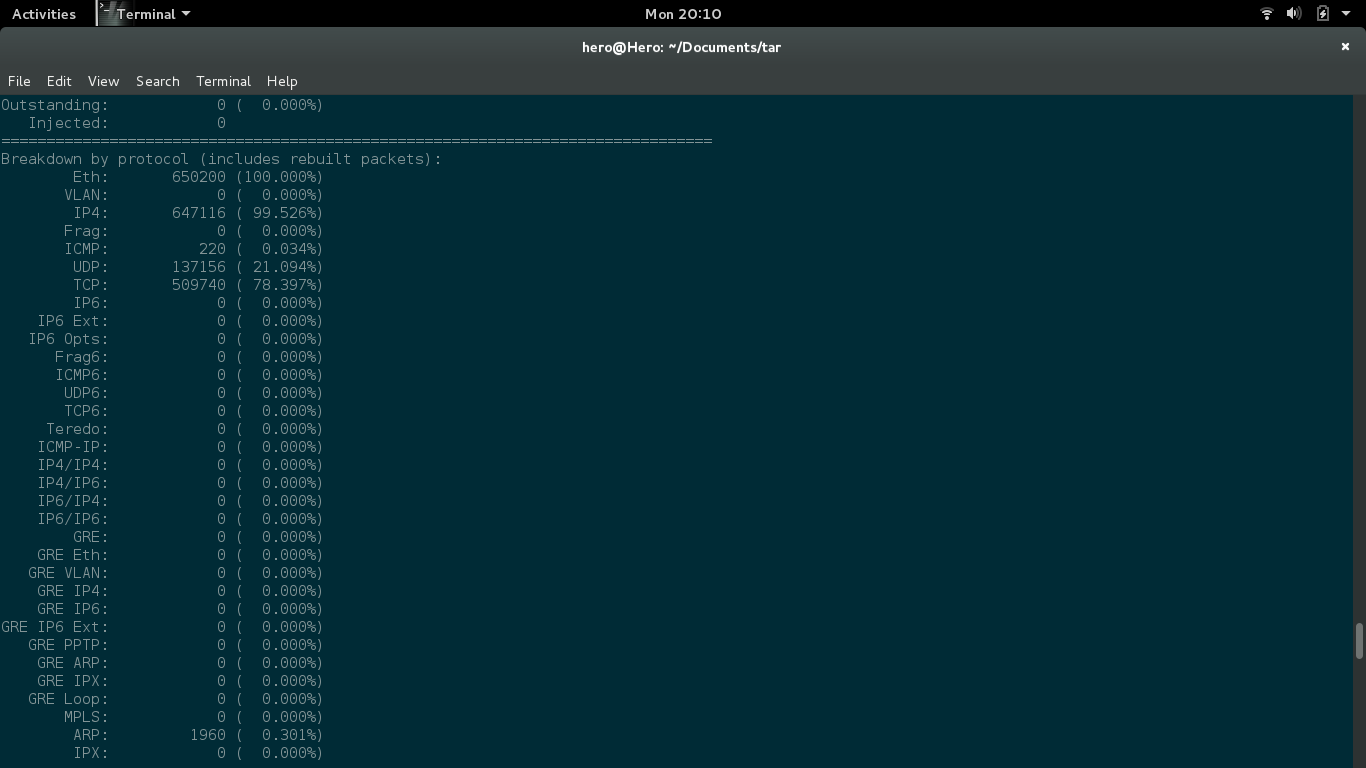


图13

最后输入alert占比，如下图14所示：

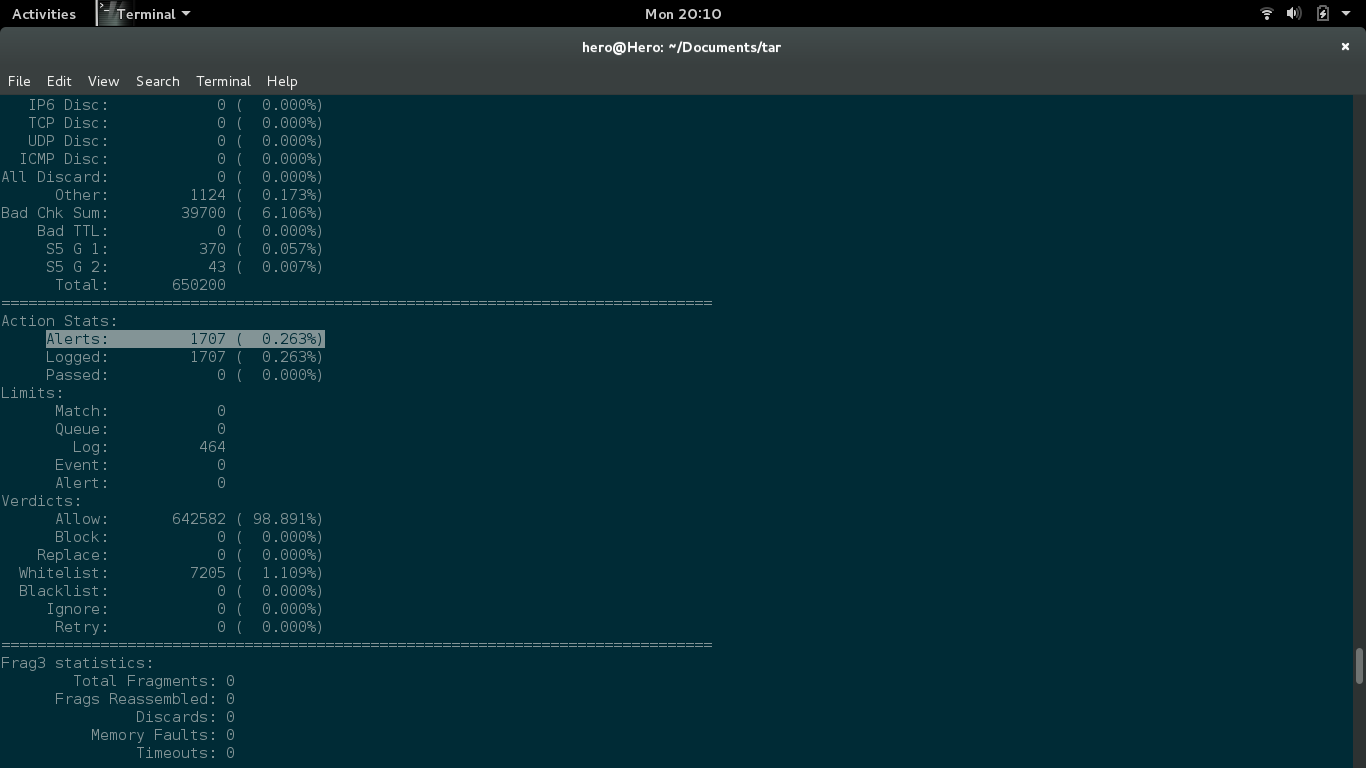


图14

现在可以查看一下Snort的输出alert文件，如下图15所示。

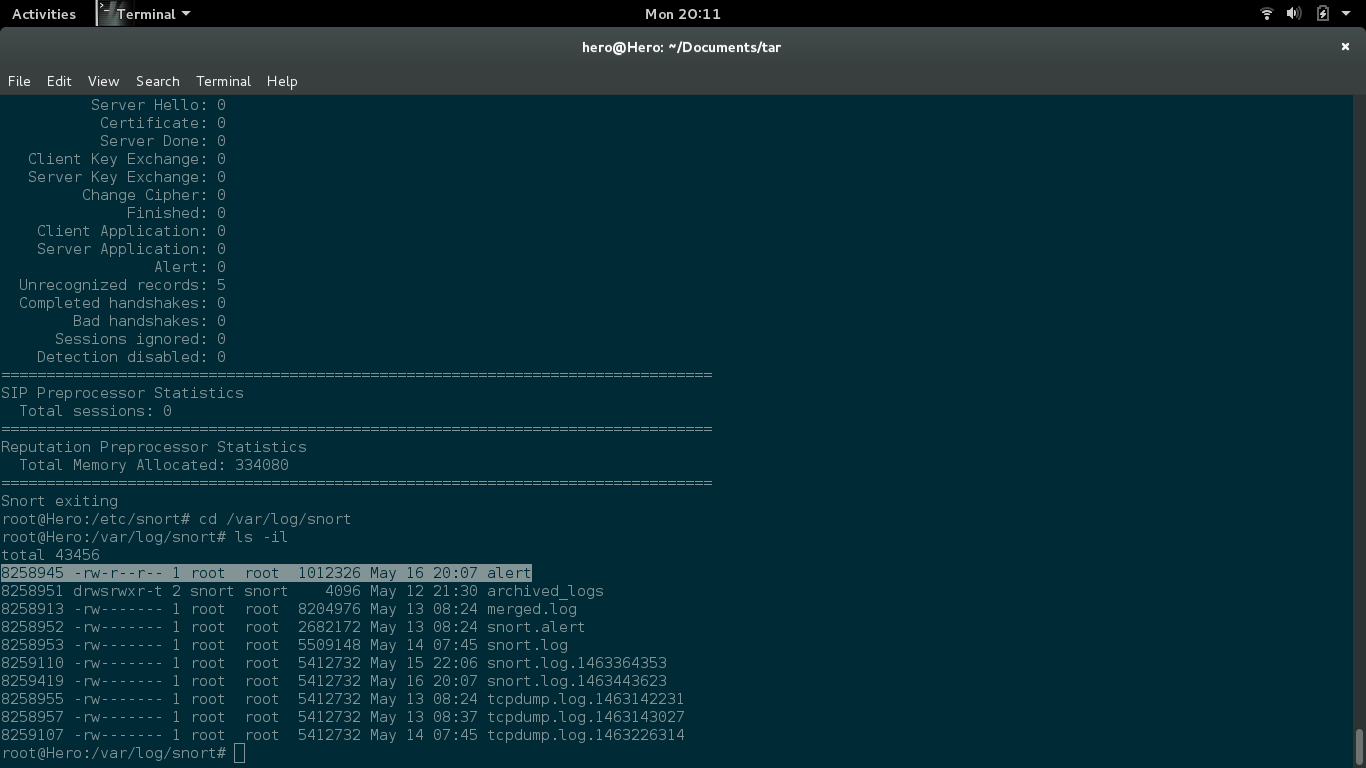


图15

打开alert文件，如下图16所示:

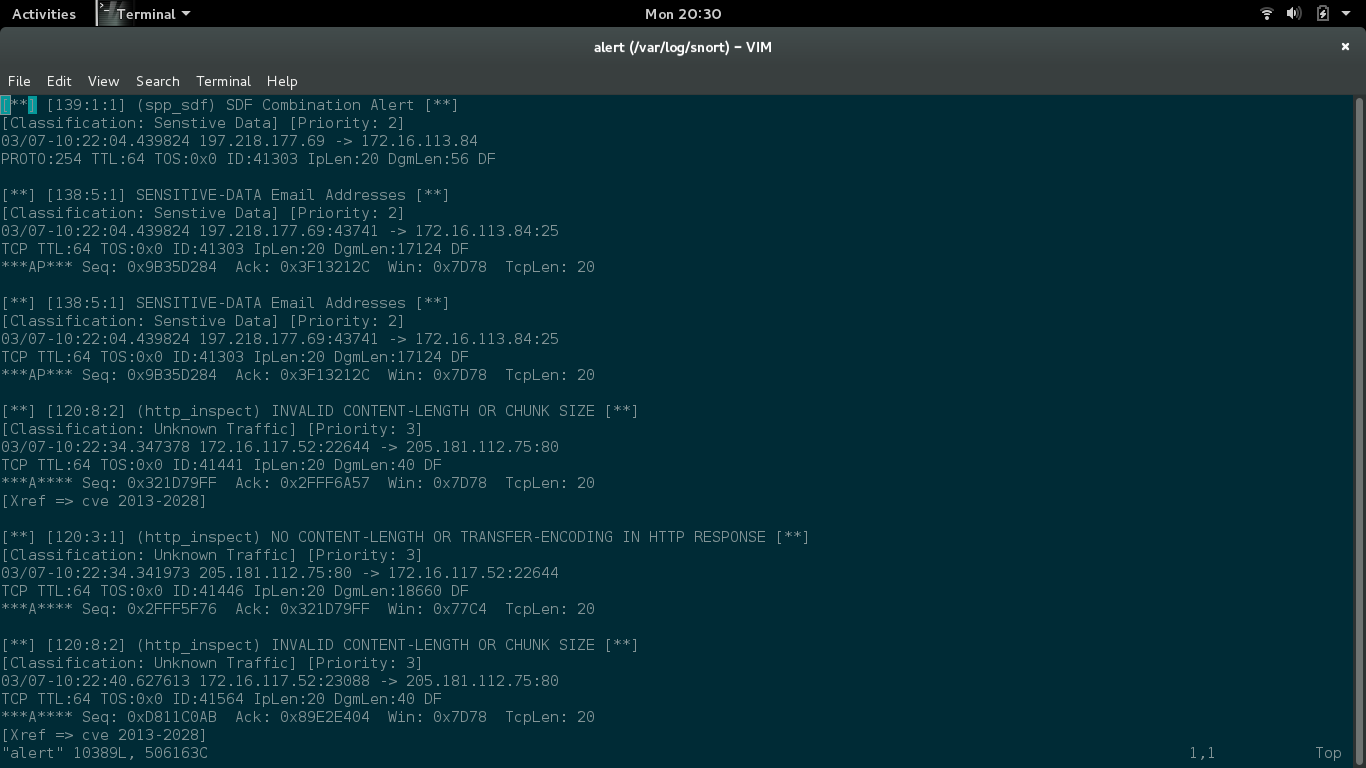


图16

到这里我们已经使用Snort对dump文件进行了初步的处理，下面可以进行进一步的分析。

（2）告警日志的分析和预处理

日志中存在太多的无用信息，如：

03/07-22:22:04.439824 [\*\*] [139:1:1] (spp\_sdf) SDF Combination Alert [\*\*] [Classification: Sensitive Data] [Priority: 2] {PROTO:254} 197.218.177.69 -> 172.16.113.84

其中，SDF Combination Alert代表Snort本身的保卫系统的告警。Senstive Data代表这个流量是敏感数据流量。

现在我们首先去除其中不需要的各种符号，通过一个简单的Python程序实现，处理后的数据，如：

03/07-22,22, 04.439824, [139,1,1], spp\_sdf SDF Combination Alert , [Classification, Senstive Data], [Priority,2], PROTO, 254, 197.218.177.69 , 172.16.113.84

然后，我们通过分析日志可以看出，时间上是存在连续性的，所有时间本身也是对关联分析是没有任何意义的，去除时间和杂项后的数据，如：

spp\_sdf SDF Combination Alert, Senstive Data, 197.218.177.69, 172.16.113.84

现在我们就可以还原攻击现场，官方背景说明了LLDOS数据集是一个典型的DDOS攻击场景，DDOS攻击的目标IP只会是一个，但是攻击IP可以是很多个。

（3）LLDOS\_1.0\_inside数据的关联分析和可视化

①对1.0版本的inside内网的数据使用目的IP分篮进行关联分析：

注：IP地址后的s标记和d标记是表明该IP是源ip（source IP）还是目的IP（destination IP）。

使用目的IP作为分篮依据且最小支持度为0.1情况下的关联结果（有删减）：

[u'172.16.116.44s'], 18

[u'SENSITIVE DATA Email Addresses', u'Misc activity', u'Senstive Data', u'spp\_sdf SDF Combination Alert'], 18

[u'172.16.113.168s'], 19

[u'spp\_sdf SDF Combination Alert', u'Unknown Traffic'], 19

[u'Senstive Data', u'spp\_sdf SDF Combination Alert', u'Unknown Traffic'], 19

[u'http\_inspect NO CONTENT LENGTH OR TRANSFER ENCODING IN HTTP RESPONSE', u'Senstive Data', u'spp\_sdf SDF Combination Alert'], 19

[u'172.16.117.132s'], 19

[u'Misc activity', u'Senstive Data', u'spp\_sdf SDF Combination Alert'], 19

[u'202.77.162.213s', u'Information Leak', u'PROTOCOL ICMP Unusual PING detected', u'Misc activity'], 20

[u'202.77.162.213s', u'Information Leak', u'PROTOCOL ICMP Unusual PING detected', u'PROTOCOL ICMP PING'], 20

[u'Misc activity', u'Attempted Administrator Privilege Gain'], 20

[u'172.16.113.204s', u'http\_inspect INVALID CONTENT LENGTH OR CHUNK SIZE', u'Unknown Traffic'], 22

[u'PROTOCOL ICMP Unusual PING detected', u'PROTOCOL ICMP PING', u'Misc activity'], 23

[u'Information Leak', u'PROTOCOL ICMP Unusual PING detected', u'PROTOCOL ICMP PING'], 23

[u'Information Leak', u'PROTOCOL ICMP PING', u'Misc activity'], 23

[u'SERVER MAIL Metamail header length exploit attempt', u'SENSITIVE DATA Email Addresses', u'Senstive Data', u'spp\_sdf SDF Combination Alert'], 23

[u'SERVER MAIL Metamail header length exploit attempt', u'SENSITIVE DATA Email Addresses', u'Attempted Administrator Privilege Gain', u'Senstive Data'], 23

[u'SERVER MAIL Metamail header length exploit attempt', u'SENSITIVE DATA Email Addresses', u'Attempted Administrator Privilege Gain', u'Senstive Data', u'spp\_sdf SDF Combination Alert'], 23

[u'172.16.113.204s', u'Unknown Traffic'], 27

[u'Misc activity'], 28

[u'Attempted Administrator Privilege Gain'], 30

[u'172.16.113.148s', u'http\_inspect INVALID CONTENT LENGTH OR CHUNK SIZE', u'Unknown Traffic'], 31

[u'spp\_sdf SDF Combination Alert'], 31

[u'Senstive Data'], 31

[u'Senstive Data', u'spp\_sdf SDF Combination Alert'], 31

[u'172.16.113.204s'], 32

[u'172.16.113.148s', u'Unknown Traffic'], 36

[u'172.16.113.148s'], 43

[u'http\_inspect INVALID CONTENT LENGTH OR CHUNK SIZE', u'Unknown Traffic'], 126

[u'Unknown Traffic'], 145

可以发现202.77.162.213存在ICMP不正常PING探测（PROTOCOL ICMP Unusual PING detected），172.16.113.204存在无效的http访问流量（http\_inspect INVALID CONTENT LENGTH OR CHUNK SIZE），172.16.113.148也存在无效的http访问流量。由此我们可以根据目前掌握的结果，大概构建攻击图，202.77.162.213作为攻击主机，而172.16.113.204和172.16.113.148作为攻击主机的肉鸡。分析结论可以很好的和官方的背景相互印证。

②对使用源IP分篮的数据进行关联分析

在最小支持度为0.1的前提下（有删减）：

[u'spp\_sdf SDF Combination Alert'], 83

[u'spp\_sdf SDF Combination Alert', u'Senstive Data'], 83

[u'Senstive Data'], 84

[u'http\_inspect NO CONTENT LENGTH OR TRANSFER ENCODING IN HTTP RESPONSE'], 166

[u'http\_inspect NO CONTENT LENGTH OR TRANSFER ENCODING IN HTTP RESPONSE', u'Unknown Traffic'], 166

[u'Unknown Traffic'], 181

[u"IPV4 packet from 'current net' source address", u'Misc activity', u'131.84.1.31d'], 262

[u"IPV4 packet from 'current net' source address", u'131.84.1.31d'], 262

[u"IPV4 packet from 'current net' source address"], 263

[u"IPV4 packet from 'current net' source address", u'Misc activity'], 263

[u'Misc activity', u'131.84.1.31d'], 265

[u'Potentially Bad Traffic', u'131.84.1.31d'], 265

[u'Bad Traffic Loopback IP'], 265

[u'Bad Traffic Loopback IP', u'Potentially Bad Traffic'], 265

[u'Bad Traffic Loopback IP', u'Potentially Bad Traffic', u'131.84.1.31d'], 265

[u'Bad Traffic Loopback IP', u'131.84.1.31d'], 265

[u'Potentially Bad Traffic'], 276

[u'Misc activity'], 295

[u'131.84.1.31d'], 530

由上面的结果我们可以分析出，131.84.1.31存在被攻击的微信，且攻击类型是糟糕的传输环回IP（Bad Traffic Loopback IP）和潜在的坏传输（Potentially Bad Traffic），也就是大量流量短时间堆积在这个IP，导致了DDOS攻击。

③对包含202.77.162.213s告警日志的分析

我们首先提取出这个IP作为目的IP时的告警日志，进行关联分析，然后进行可视化标记。

关联分析结果（有删减）：

[u'172.16.112.105s', u'PROTOCOL ICMP Destination Unreachable Port Unreachable', u'202.77.162.213d'], 12

[u'172.16.112.100s', u'PROTOCOL ICMP Destination Unreachable Port Unreachable', u'202.77.162.213d'], 12

[u'172.16.112.194s', u'PROTOCOL ICMP Destination Unreachable Port Unreachable', u'202.77.162.213d'], 40

可以看到有三个IP对其返回了ICMP报文不可达的信息，现在提取出作为源IP时的告警数据（有删减）：

09hour,50mim,PROTOCOL SERVICES rsh root,202.77.162.213s,172.16.115.20d

09hour,50mim,PROTOCOL SERVICES rsh root,202.77.162.213s,172.16.115.20d

09hour,50mim,PROTOCOL SERVICES rsh root,202.77.162.213s,172.16.112.10d

09hour,50mim,PROTOCOL SERVICES rsh root,202.77.162.213s,172.16.112.50d

09hour,51mim,PROTOCOL ICMP Unusual PING detected,202.77.162.213s,172.16.115.5d

09hour,51mim,PROTOCOL ICMP PING,202.77.162.213s,172.16.115.5d

09hour,51mim,PROTOCOL ICMP Unusual PING detected,202.77.162.213s,172.16.115.20d

09hour,51mim,PROTOCOL ICMP PING,202.77.162.213s,172.16.115.20d

09hour,51mim,PROTOCOL ICMP Unusual PING detected,202.77.162.213s,172.16.115.87d

09hour,51mim,PROTOCOL ICMP PING,202.77.162.213s,172.16.115.87d

09hour,51mim,PROTOCOL ICMP Unusual PING detected,202.77.162.213s,172.16.115.234d

09hour,51mim,PROTOCOL ICMP PING,202.77.162.213s,172.16.115.234d

09hour,51mim,PROTOCOL ICMP Unusual PING detected,202.77.162.213s,172.16.113.50d

09hour,51mim,PROTOCOL ICMP PING,202.77.162.213s,172.16.113.50d

（以下删减）

可以看出202.77.162.213IP在09:50时候分别对172.16.115.20、172.16.112.10和172.16.112.50进行了rsh root攻击。rsh是“remote shell”（远程 shell）的缩写，该命令在指定的远程主机上启动一个shell并执行用户在rsh命令行中指定的命令；如果用户没有给出要执行的命令，rsh就用rlogin命令使用户登录到远程机上。

然后在09:51时候，主机202.77.162.213开始对整个网络进行PING扫描，判断存活主机。

下一步进行可视化处理和标记，如下图17所示：

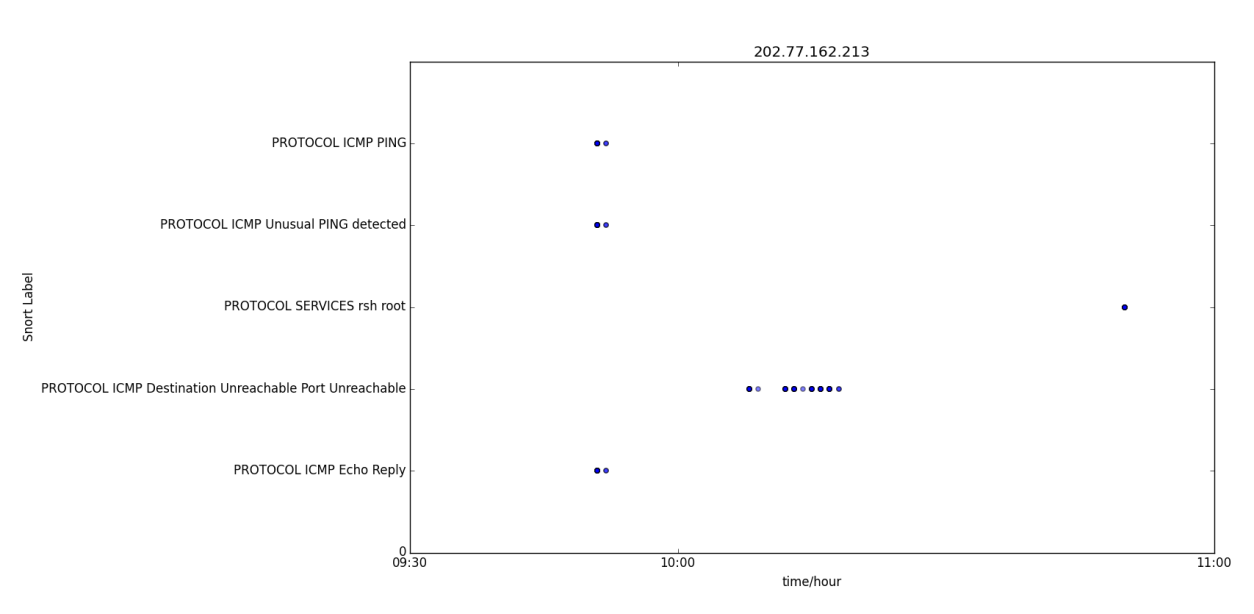


图17 202.77.162.213告警事件时间表

从图17中可以看出，对该IP的告警事件主要有五个类型，其中，PROTOCOL ICMP PING、PROTOCOL ICMP Unusual PING detected、PROTOCOL ICMP Destination Unreachable Port Unreachable和[ROTOCOL ICMP Echo Reply都是属于对目标的扫描，而PROTOCOL SERVICES rsh root属于对目标攻击、远程登录和操作。从图中还可以看出，在接近10点的时候，主机开始扫描内网，10点多的时候，开始攻击内网中的某些主机，由此我们可以推测攻击者的IP就是202.77.162.213。

现在我们给出攻击的强度图，强度越大的，亮度越高，如下图18所示：

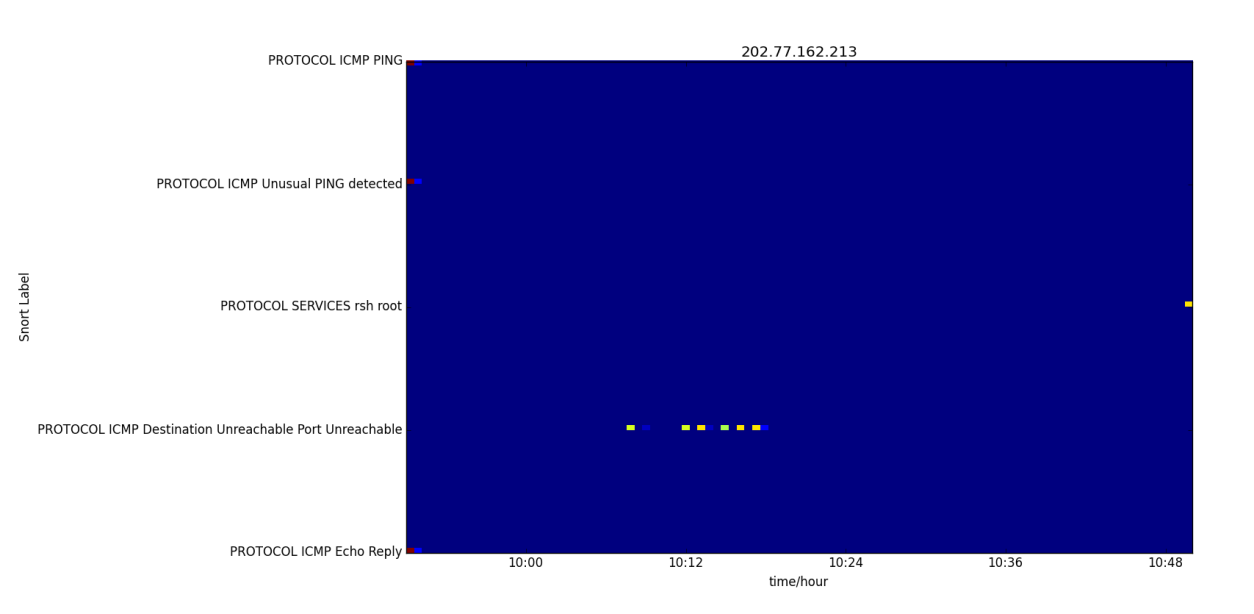


图18 202.77.162.213告警事件强度图

明显的可以看出，10:00到10:24之间的不可达告警（ PROTOCOL ICMP Destination Unreachable Port Unreachable）的坐标位置有一条很明亮的条纹，其亮度明显比icmp回应（PROTOCOL ICMP Echo Reply）的亮，意味着内网中的许多主机是处于不开机状态的，其数量是多于开机的数量。在x轴上10:48的位置，有个明显的亮点，说明rsh root其攻击强度也是比较强的。

④对172.16.113.148告警日志的分析

首先对数据进行预处理，然后进行可视化处理和标记，处理和标记的结果如下图19所示：

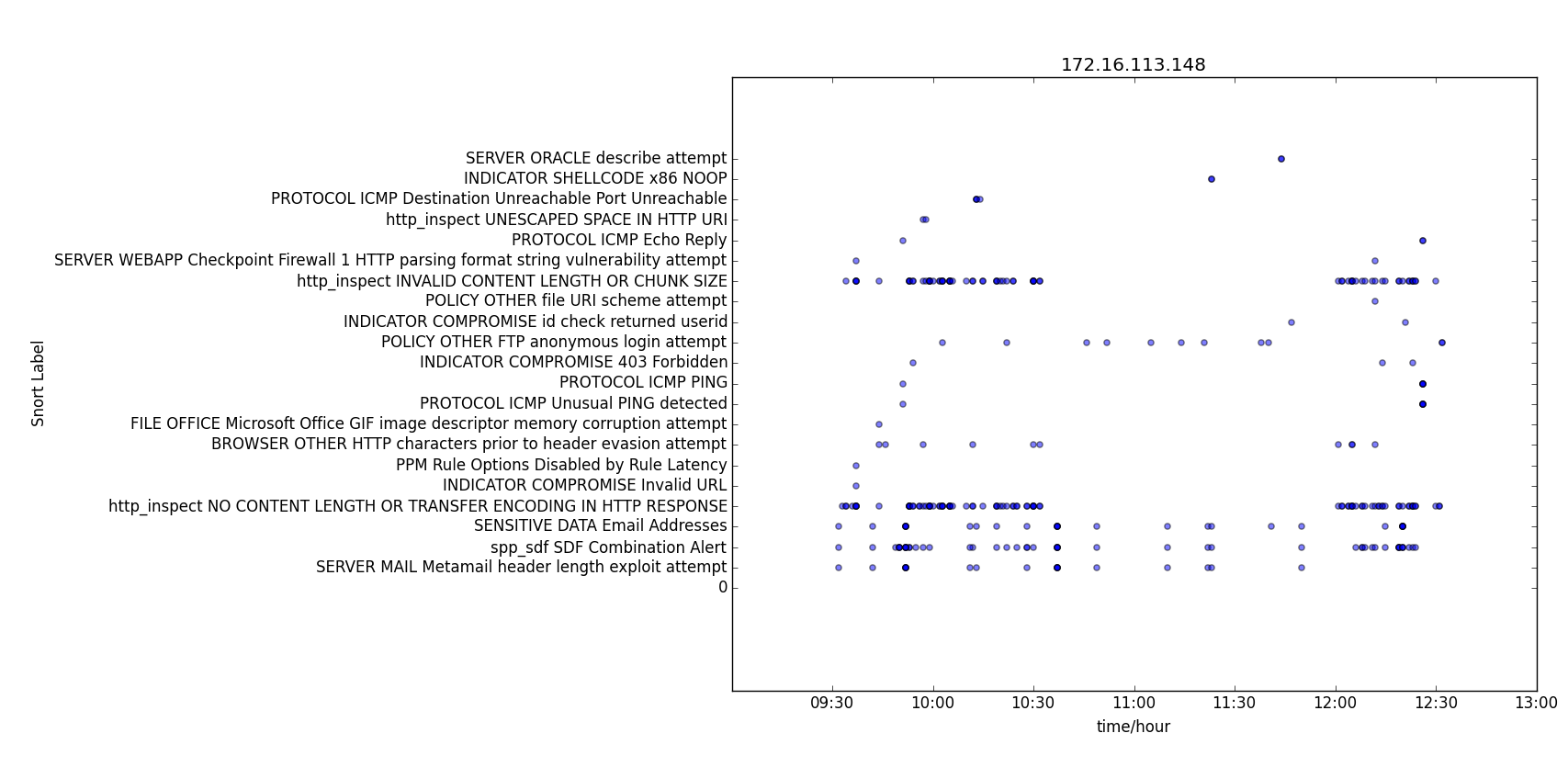


　　　　　　　　　图19 172.16.113.148告警事件时间表

可以清楚的看到172.16.113.148的主要告警事件为http\_inspect类型的，包括http\_inspect INVALID CONTENT LENGTH OR CHUNK SIZE和http\_inspect NO CONTENT LENGTH OR TRANSFER ENCODING IN HTTP RESPONSE，在接近11:30附近时刻，并未出现大量的报文，所有肉鸡很有可能不是这个主机。

下面的图20给出了攻击的能量分析：

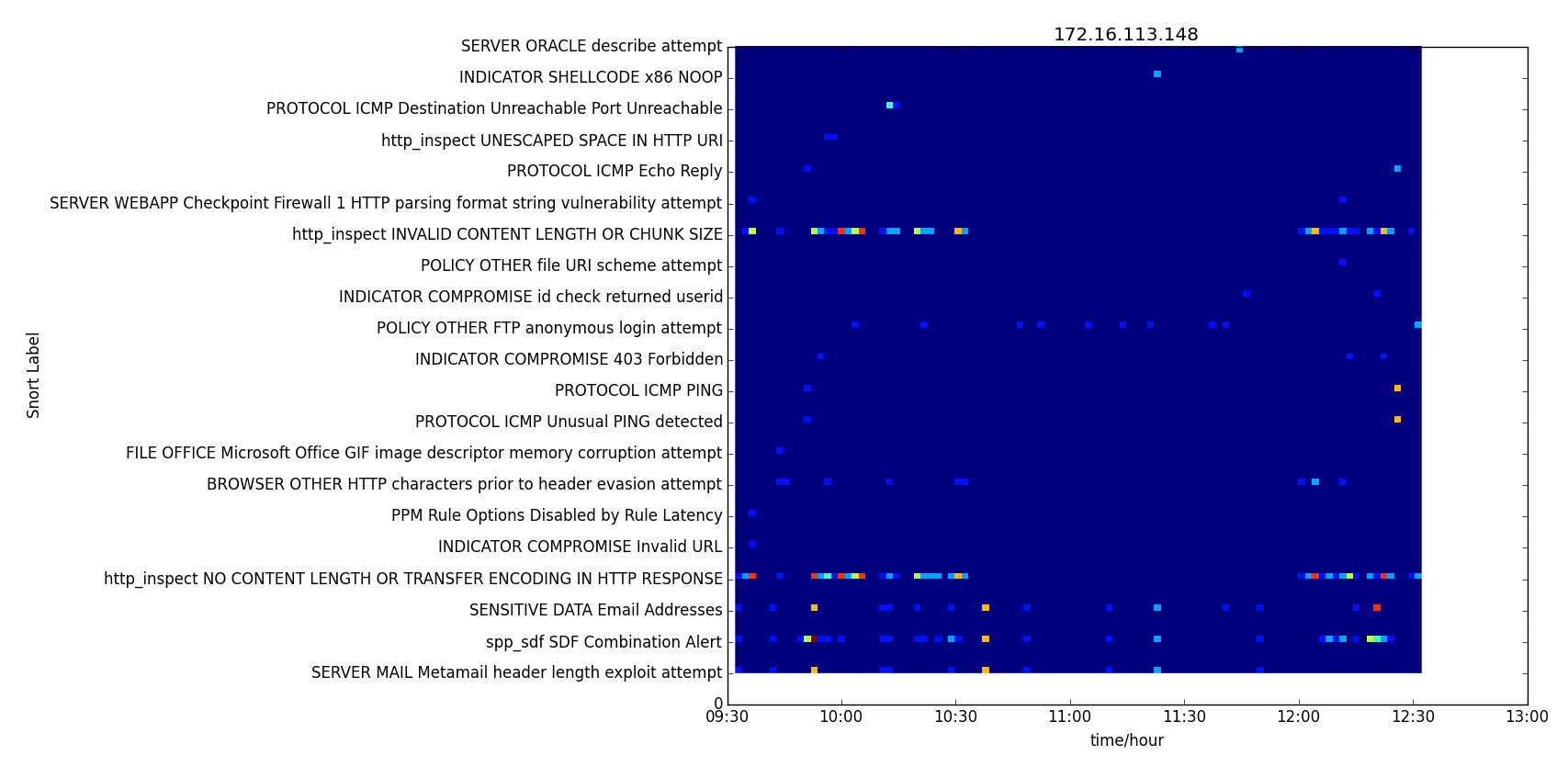


图20 172.16.113.148告警事件强度图

可以看出，http\_inspect类型的亮度也是最强的，而且http\_inspect INVALID CONTENT LENGTH OR CHUNK SIZE和http\_inspect NO CONTENT LENGTH OR TRANSFER ENCODING IN HTTP RESPONSE都是同时出现，同时消失的。

⑤对172.16.113.204告警日志的分析

同样对数据进行预处理和分析，然后进行可视化处理和标记，如下图21所示：

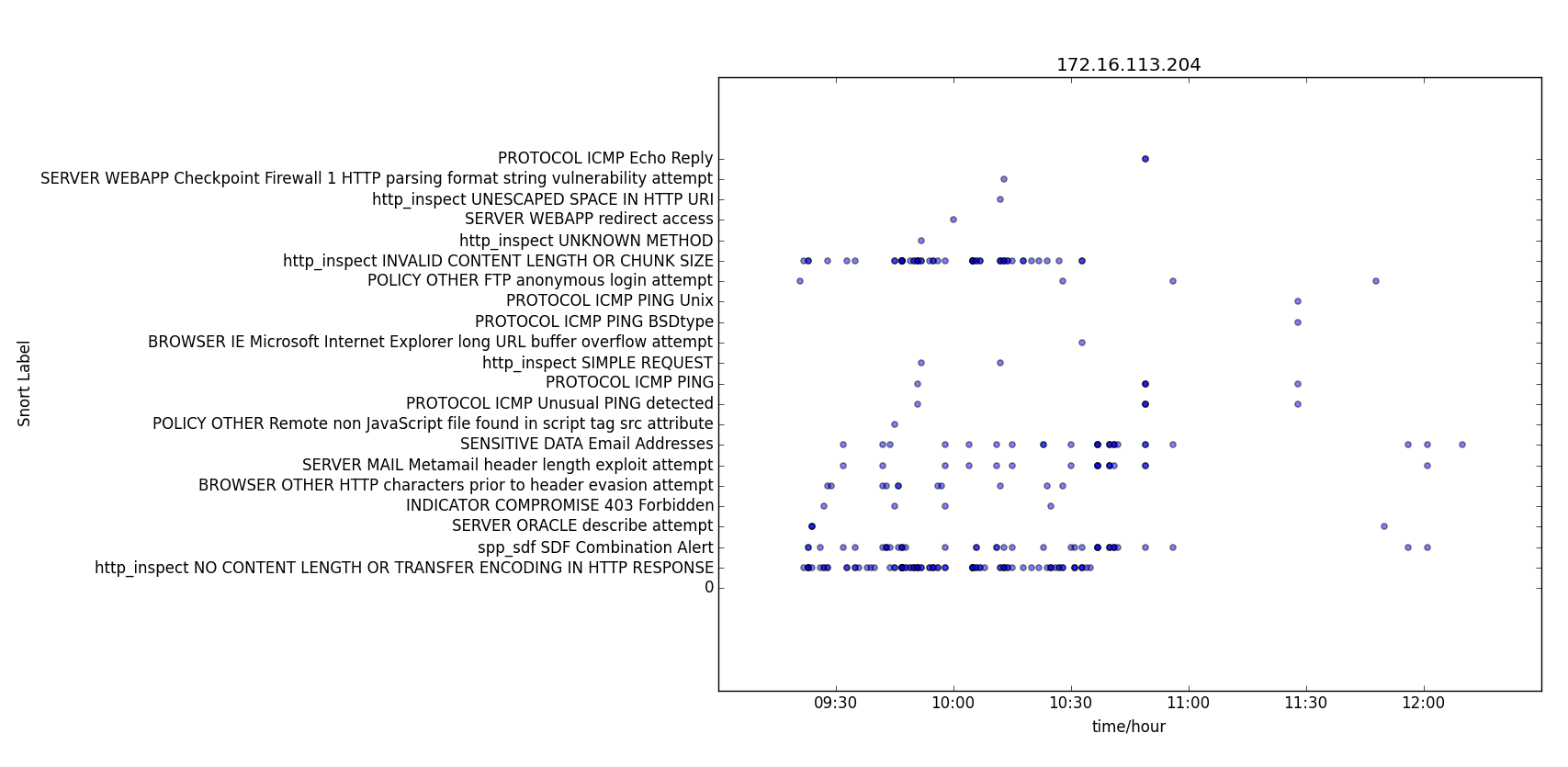


　　　　　　　　图21　172.16.113.204告警事件时间表

从图中可以发现，172.16.113.204主机只有前端的告警数据，没有11:30附近的明显的告警数据，所以这个主机很有可能不是肉鸡。

下图22给出172.16.113.204攻击的能量分析：

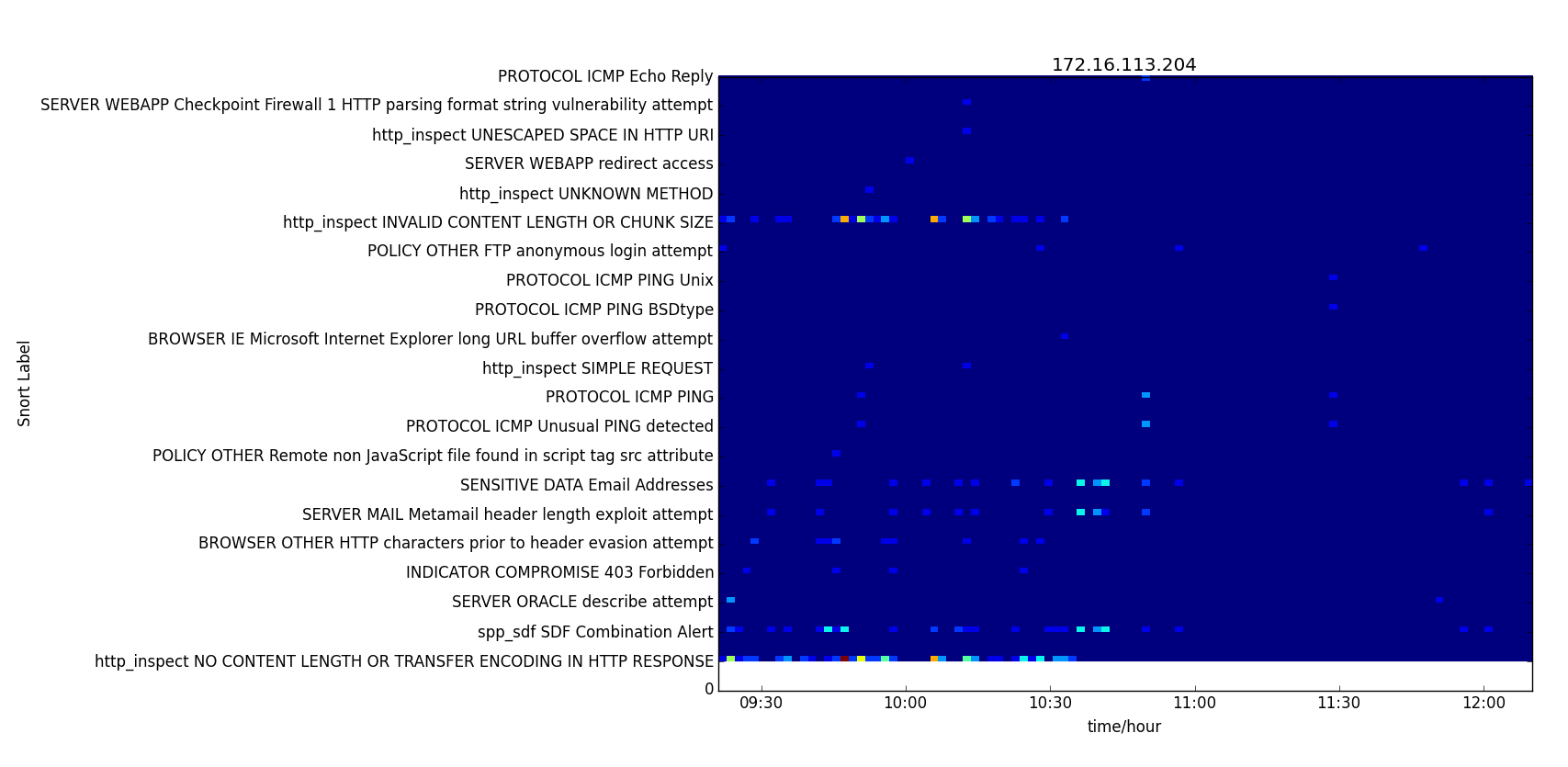


图22 172.16.113.204告警事件强度图

图中所示的亮度也出现在前面的时间段里，所以其并不是肉鸡。

⑥对172.16.115.20告警日志的分析

同样对数据进行预处理和分析，然后进行可视化处理和标记，如下图23所示：

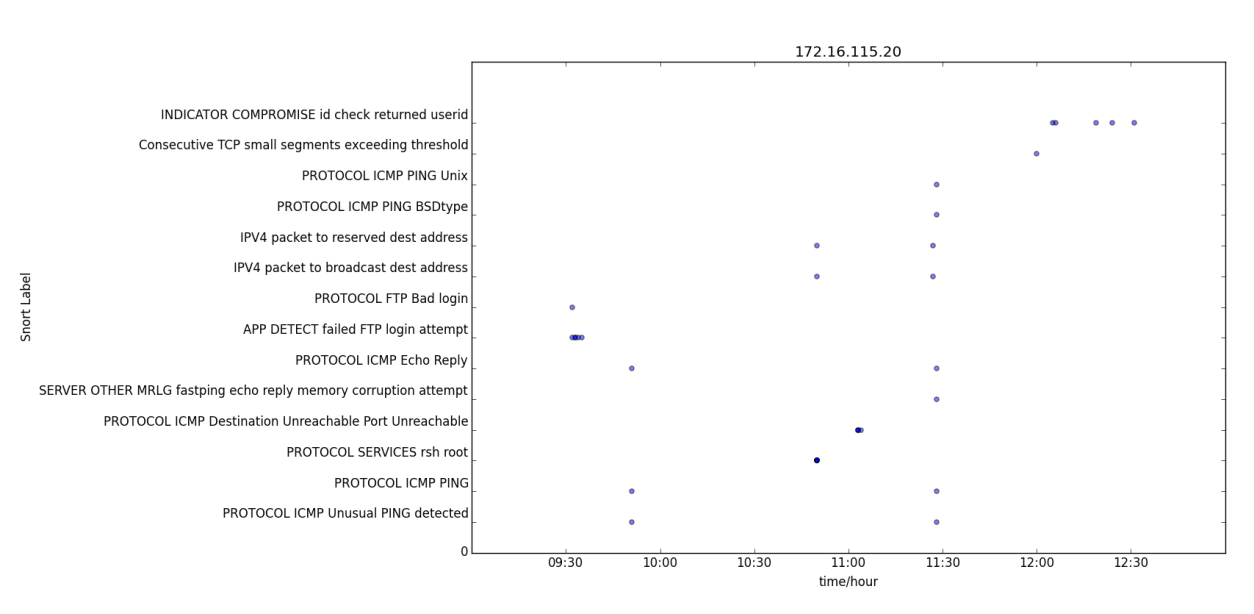


图23 172.16.115.20告警事件时间表

从图中不难看出，在11:00时刻附近，有一处远程root控制（RPOTOCOL SERVICES rsh root）告警，这时的主机遭受了漏洞攻击，并夺取了主机的控制权。而在11:30附近，则出现了整齐的报文包发送，正常主机是不会同时使用到这么多不同种类的ICMP协议的，这里面包括了PROTOCOL ICMP Unusual PING detected、PROTOCOL ICMP PING、SERVER OTHER MRLG fastping echo reply memory corruption attempt、PROTOCOL ICMP Echo Reply、IPV4 packet to broadcast dest address、IPV4 packet to reserved dest address、PROTOCOL ICMP PING BSDtype、PROTOCOL ICMP PING Unix八种不同类型，由此分析，其很有可能在实施DDOS攻击。

下面的图24给出告警事件强度：

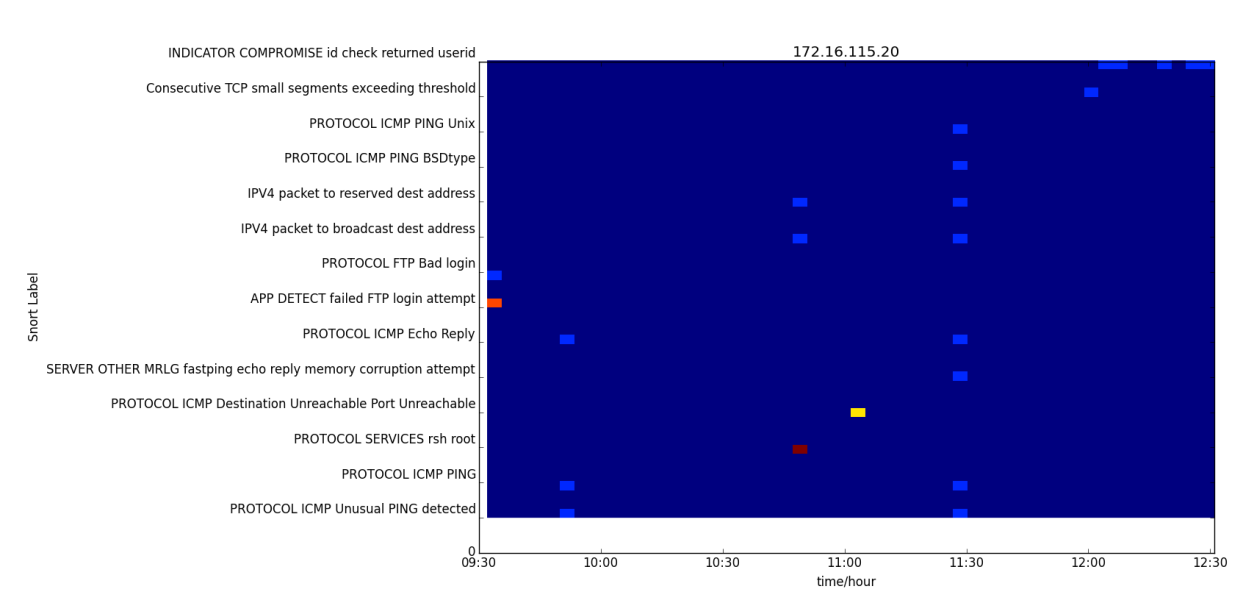


图24 172.16.115.20告警事件强度图

从中可以看到几个比较明亮的点，其中09:30附近的亮度由于刚开始，而且告警类型和DDOS没有太大关系，自动忽略。其他的主要有一个11:00附近的PROTOCOL SERVICES rsh root告警和PROTOCOL ICMP Destination Unreachable Prot Unreachable告警。

⑦对172.16.112.10告警日志的分析

同样对数据进行预处理和分析，然后进行可视化处理和标记，如下图25所示：

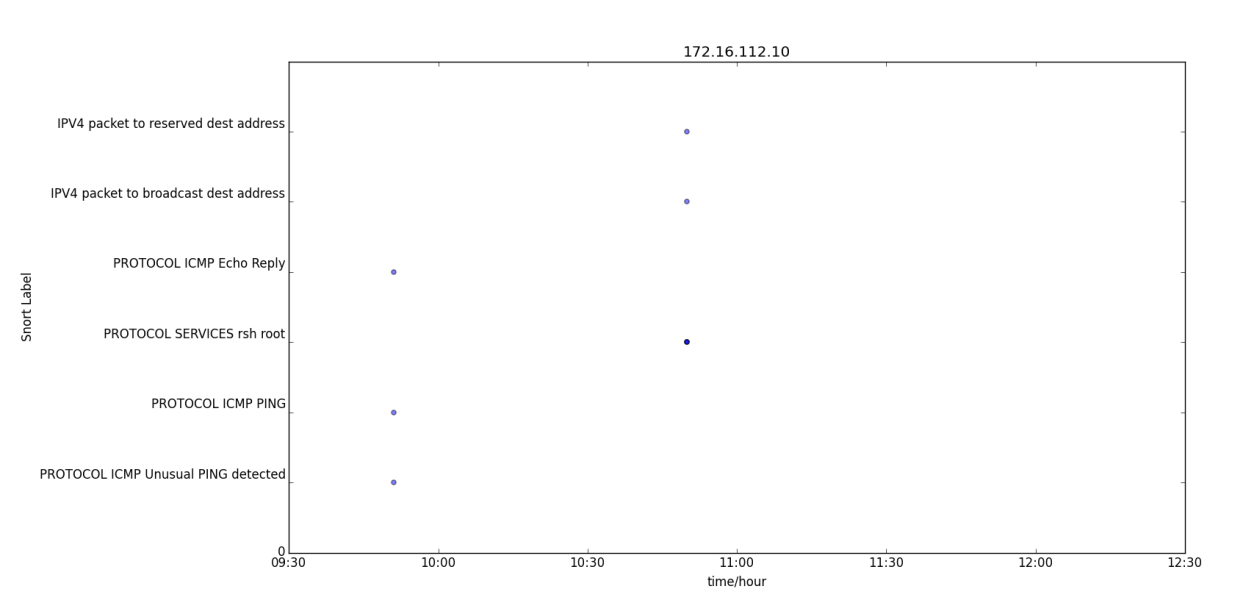


图25 172.16.112.10告警事件时间表

从图中可以很清晰的发现在11:00附近，有个PROTOCOL SERVICES rsh root告警，强度图由于数据量太小无法分析所以将不给出。

⑧对172.16.112.50告警日志的分析

同样对数据进行预处理和分析，然后进行可视化处理和标记，如下图26所示：

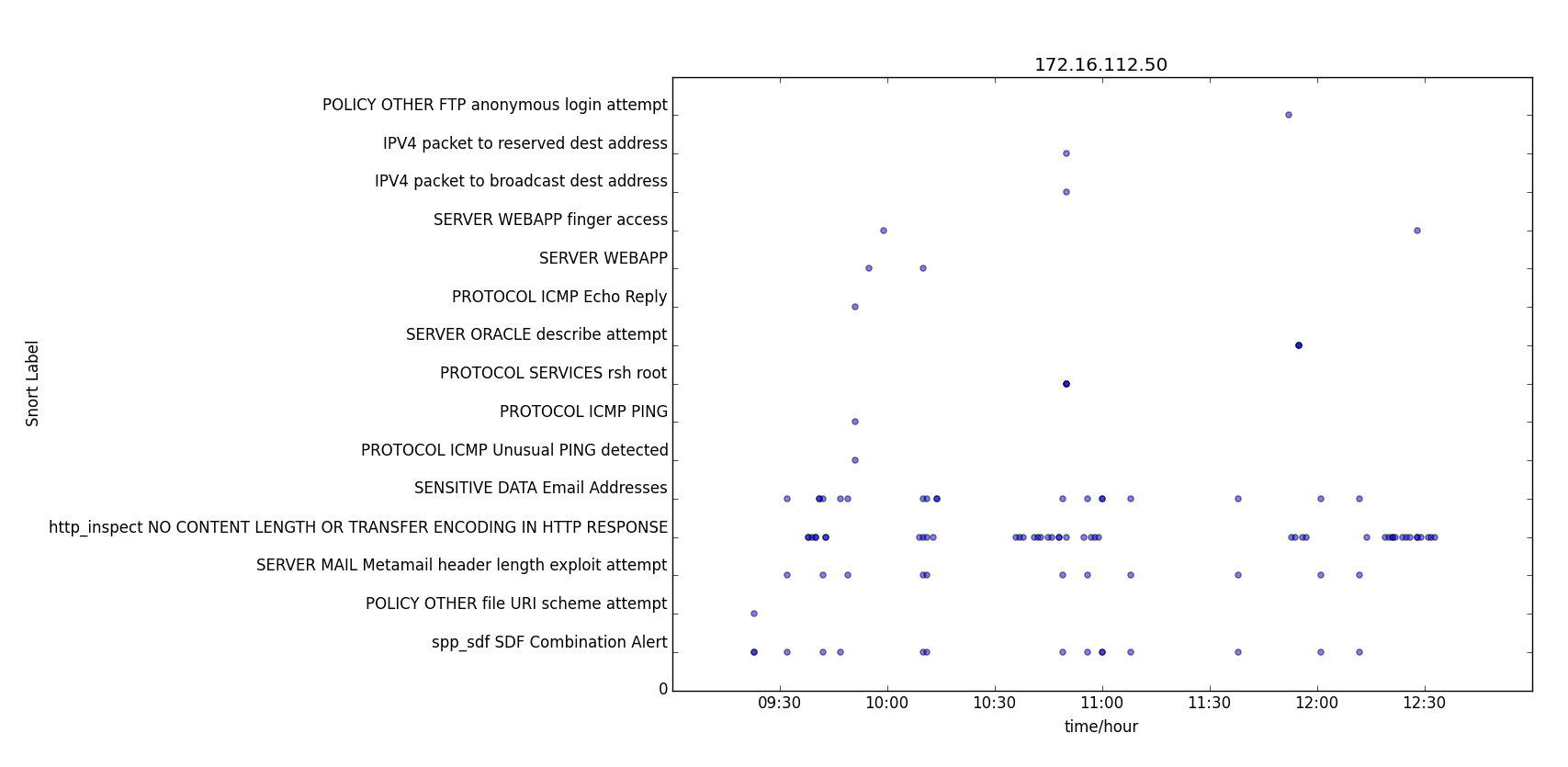


图26 172.16.112.50告警事件时间表

从这个图中看不出什么异常。

下面的图27给出172.16.112.50的告警强度：

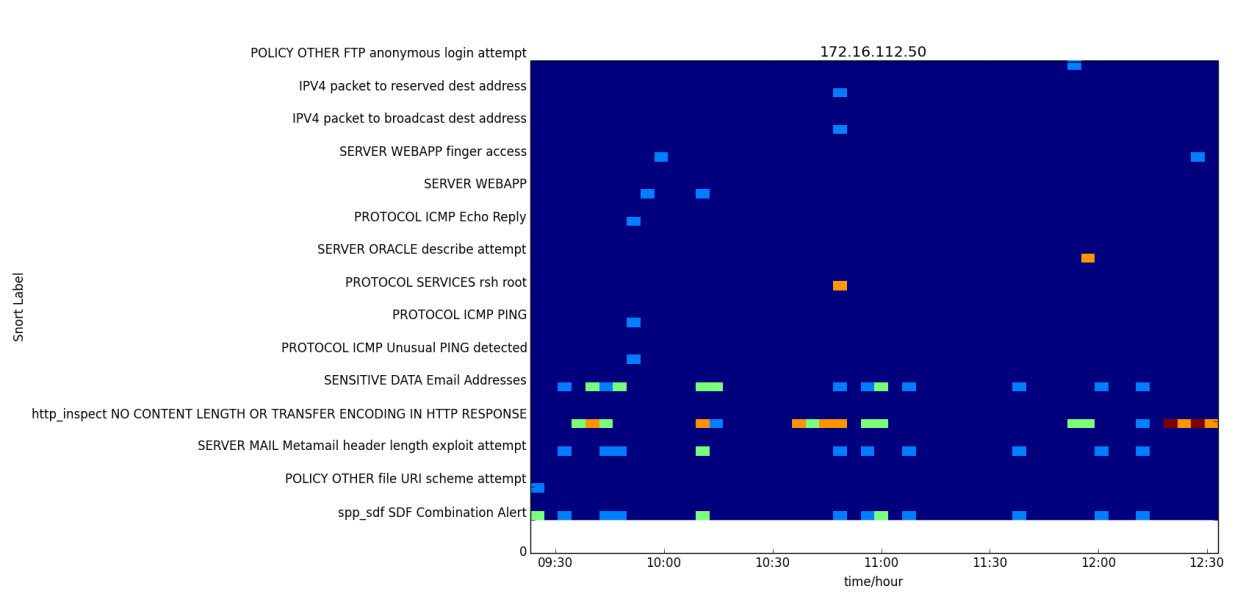


图27 172.16.112.50告警强度图

亮度明显比较亮的是http\_inspect NO CONTENT LENGTH TRANSFER ＥＮＣＯＤＩＮＧ IN ＨＴＴＰ RESPONSE告警。

⑨对131.84.1.31（被攻击主机）告警日志的分析

同样对数据进行预处理和分析，然后进行可视化处理和标记，如下图28所示：

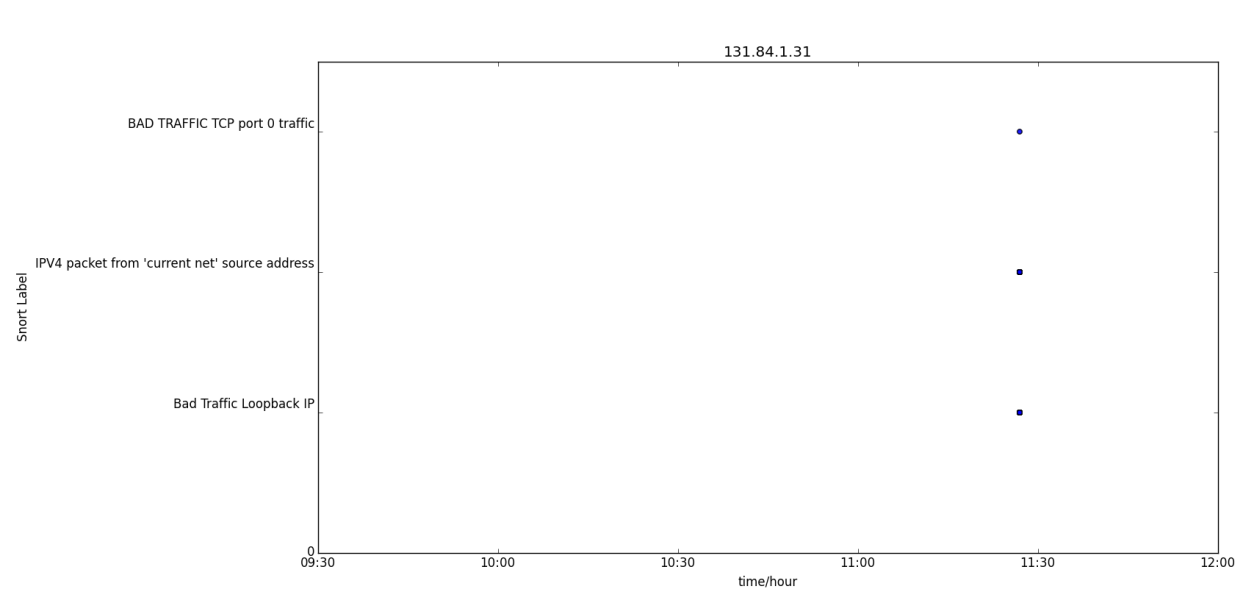


图28 131.84.1.31遭受攻击行为时间表

从图中可以清晰的看到，131.84.1.31告警日志集中在11:30左右的时候，且攻击类型也很单一，都是典型的DDOS攻击。

下面的图29给出其被攻击的能量分析：

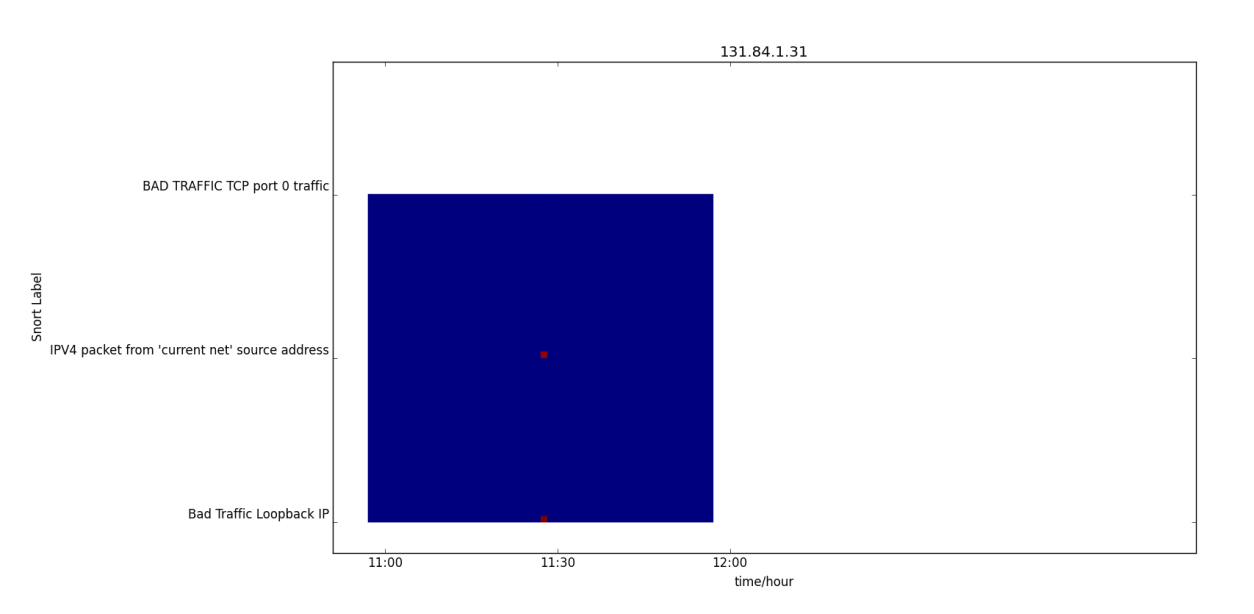


图29 131.84.1.31攻击强度图

图中明显的两个点，说明IP在内网里（IPV4 packet from current net source address）和糟糕的环回阻塞IP（Bad traffic Loopback IP）出现的频率高于糟糕的0端口的tcp坏流量（BAD TRAFFIC TCP port 0 traffic）的频率。

⑩所有告警日志分析

下面的图30至图3给出了全部告警日志的统计分析结果：

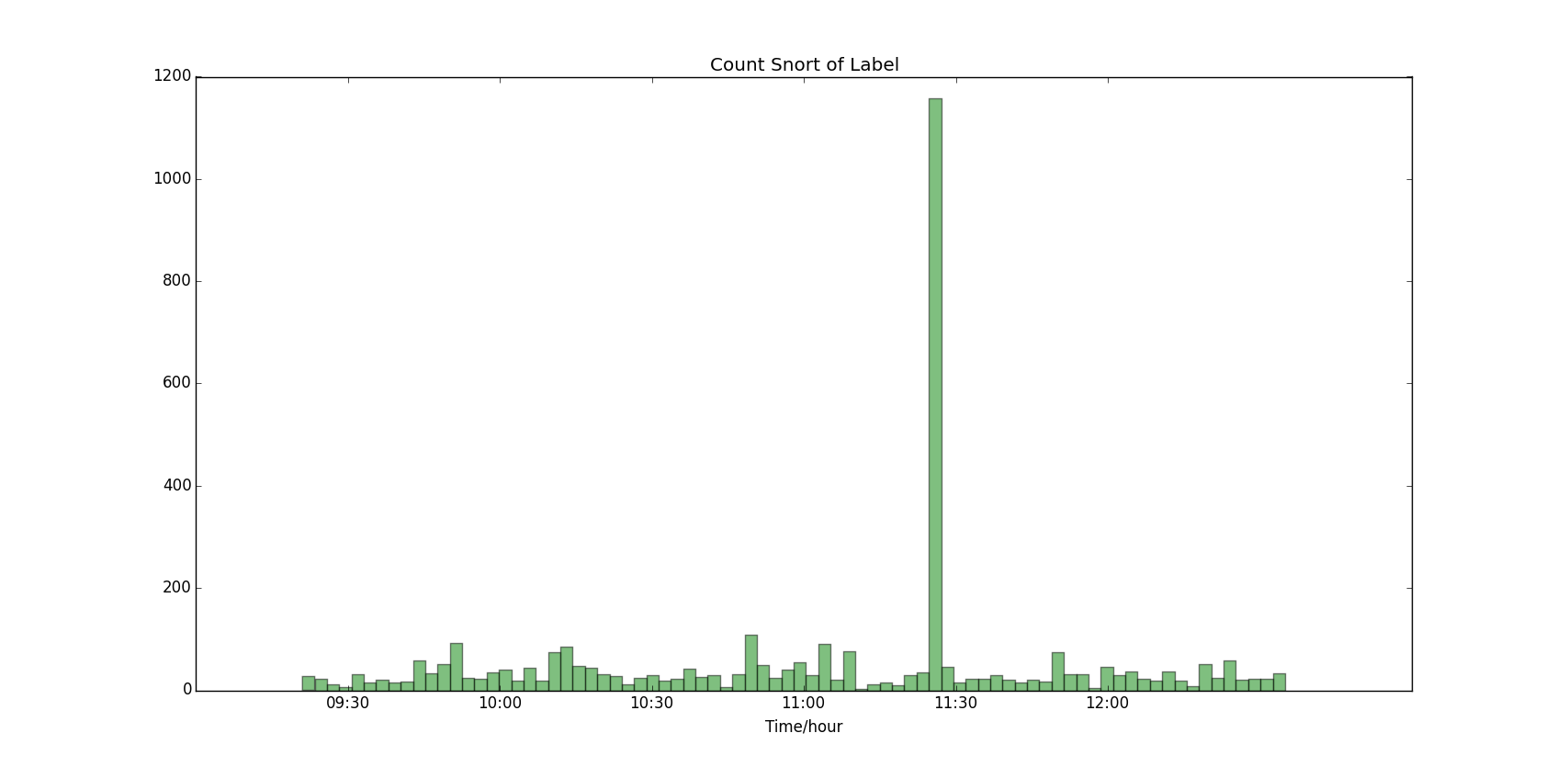


图30 所有告警日志各时段攻击的分布图

从图30中可以看出，当时间到11:30附近时刻时，告警日志突然爆发性增长，一瞬间有接近1200个告警，而其他时刻的告警平均也不超过200，明显超出了正常的告警数，因此我们断定，在11:30时刻，内网里遭受了DDOS攻击。

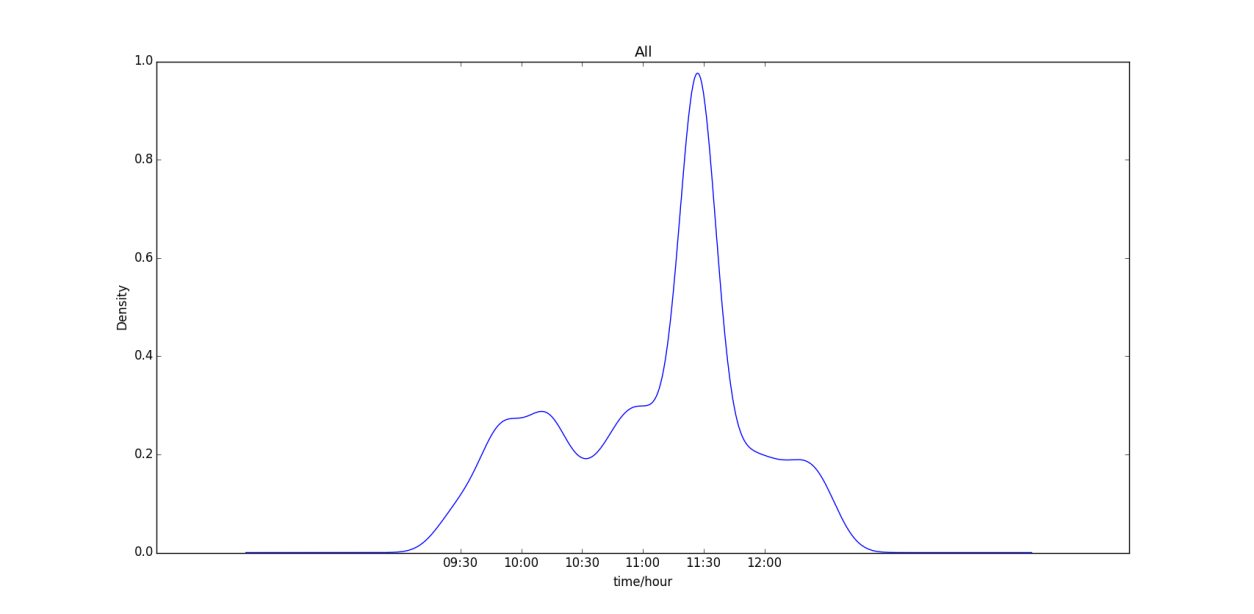


图31 所有告警日志时间的密度直方图

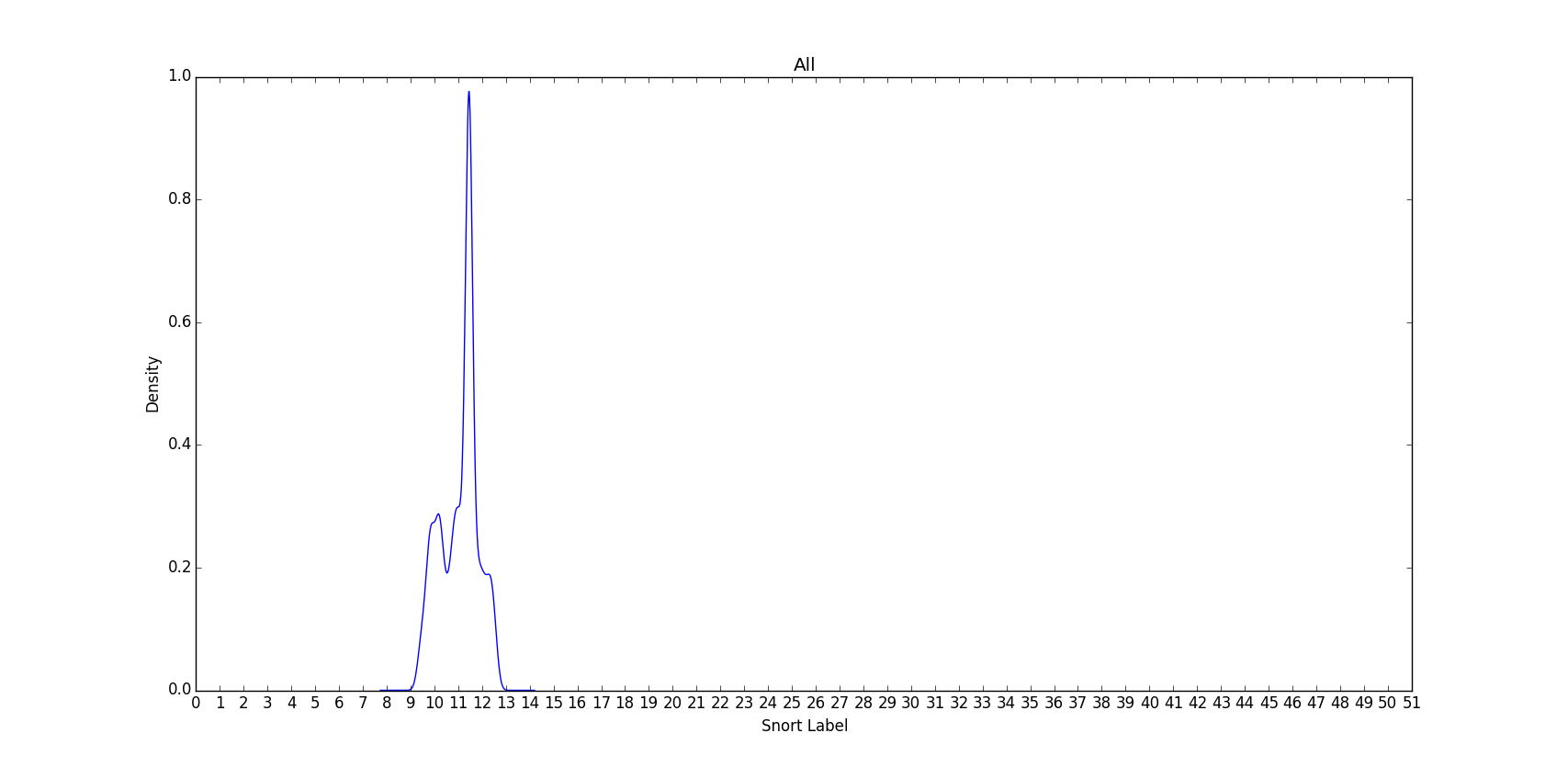


图32 所有告警日志类型的密度直方图

其中，10为Oracle服务尝试调用（SERVER ORACLE describe attempt），11为Windows NT操作系统会话（OS WINDOWS NT NULL session），12为邮件服务漏洞尝试（SERVER MAIL Metanauk header length exploit attempt）。

**4.4 本章小结**

针对现有方法的入侵证据单一，系统资源消耗大及最终结果不准确等问题，提出了一种新的攻击意图识别方法，将IDS的告警事件与数据挖掘技术相融合,构成补偿性入侵检测系统。使用FP-Growth算法从告警日志中挖掘出频繁告警；最终将产生的频繁攻击模式关联以还原攻击场景，从而推断攻击者的攻击意图和攻击目标。实验结果表明，该方法可准确识别攻击意图和攻击目标。

# 第5章 总结和展望

**5.1 论文的主要工作**

网络异常流量检测的研究工作对网络的安全和社会的稳定和谐具有重大的显示意义，为信息安全体系的建设奠定了坚实的基础。国内外对于网络异常流量检测的理论研究已经比较成熟，但转换理论为实现的工作却鲜有人做。为此，本文给出了一种基于Spark的网络异常流量检测方法，该方法有效解决了网络异常流量检测中数据量大、时效性高、计算复杂等问题。计算实例表明，应用Spark计算和处理流量数据具有快速、可靠等特点。

归纳一下，本文的工作主要包括以下几个方面：

（1）描述了APT攻击的定义，研究了国内外APT攻击研究的可靠方法，描述了开源入侵检测系统Snort框架，并使用其产生告警数据，对网络异常流量检测的一般模型进行了分析、总结。最后，叙述了网络异常流量检测的基本原理和流程。

（2）研究了K-means算法和FP-growth算法

文中分别描述了聚类分析、关联分析的基本原理，并进行了比较分析。给出了几种典型的聚类分析方法和关联分析方法，其中对K-means聚类算法和FP-growth关联分析算法进行了专门的研究。

（3）建立Spark框架下基于聚类分析的网络异常流量检测模型

通过建立基于Spark的网络异常流量检测模型，给出了网络异常流量检测的具体过程。首先，选择成熟入侵原始数据KDD-99进行分析。然后，使用Python预处理数据，包括解析数据、分割数据，最后形成可计算、可迭代的数据，并构建聚类算法指标体系。最后，使用不同参数的处理数据，带入Spark中进行聚类运算，并依据指标体系的标准，计算其准确率。对用于预处理数据的处理参数设定，给出了三种不同模式下的输出数据进行聚类，并比较其优劣。

（4）建立Spark框架下基于关联分析的网络异常流量检测模型

给出了该模型的系统设计，利用LLDOS攻击数据进行关联和分析，通过关联分析，找出入侵者和分析入侵步骤，并利用绘图进行研究攻击行为，验证了关联分析的可行性。

**5.2 存在的问题及工作展望**

本文将Spark大数据平台运用于网络异常流量检测，但该方法仍存在一些不足和需要改进的方面：

1. 并未在实现中部署分布式计算环境，并未验证在分布式计算条件下，Spark框架下不同算法间的数据处理能力和分析能力异同。
2. 对告警日志的预处理还不够细致，并不能很好的体现出告警日志的原本意义。
3. 并未使用Scala语言，未比较Python和Scala语言处理速度和质量的优劣。

# 致谢

在四年的x校生活即将结束之际，我必须感谢很多教导、帮助、关心、支持我的领导、教员和战友们，无论是生活方面，还是学习方面，他们均给予了莫大的支持和帮助，陋言片纸，难以尽表我的感激之情。

首先我要特别感谢我的指导老师吴x教授和李x博士，是他们指导我完成了大学最后的毕业论文设计。吴x教授倾尽所学帮我分析课题重点、难点，给予我相关的重要资料，李x博士在毕业设计期间，帮我分析下一步工作，以及对以前工作的回顾。在我撰写该论文的过程中，由于理论知识薄弱，对理论理解不透彻，曾多次耽误进度，是吴x教授和李x博士给予我包容和鼓励，并牺牲宝贵时间对我进行指导和帮助。没有吴x教授和李x博士的悉心指导，我不可能完成本次毕业设计。对此，我向他们表示无尽的感激和敬意！

非常感谢我大学期间的所有教员和老师，感谢他们在文化知识上的传授和教育。

感谢我大学期间一起生活过的xx和xx兄弟，感谢他们在日常生活及工作中的团结帮助与鼓励支持。

感谢我的家人对我的关心和帮助，他们无私的奉献是我一路走到今天的基础，他们的理解和支持是我永远的源动力，祝他们身体健康。

感谢评委老师们对本论文的审核，期待您们的教导和点评！

# 参考文献

[1] 张帅. 对APT攻击的检测与防御[J]. 信息安全与技术, 2011(9):125-127.

[2] 林龙成, 陈波, 郭向民. 传统网络安全防御面临的新威胁:APT攻击[J]. 信息安全与技术, 2013, 4(3):20-25.

[3] 周涛. 大数据与APT攻击检测[J]. 信息安全与通信保密, 2012(7):23-23.

[4] 刘东鑫, 刘国荣, 王帅,等. 面向企业网的APT攻击特征分析及防御技术探讨[J]. 电信科学, 2013, 29(12):158-163.

[5] 许婷. 一种有效防范APT攻击的网络安全架构[J]. 信息安全与通信保密, 2013(6):65-67.

[6] McKenna A; Hanna M; Banks E; Sivachenko A; Cibulskis K; Kernytsky A; Garimella K; Altshuler D; Gabriel S; Daly M; DePristo MA. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data[J]. Genome Research, 2014, 20(9):1297-1303.

[7] 肖志新, 杨岳湘, 杨霖. 基于小波技术的网络异常流量检测与实现[J]. 计算机科学, 2006, 33(10):116-118.

[8] 曾嘉, 金跃辉, 叶小卫. 基于NetFlow的网络异常流量检测[J]. 网络新媒体技术, 2007, 28(7):709-713.

[9] 刘勇, 香丽芸. 基于网络异常流量判断DoS/DDoS攻击的检测算法[J]. 吉林大学学报:信息科学版, 2008, 26(3):313-319.

[10] 燕发文, 黄敏, 王中飞. 基于BF算法的网络异常流量行为检测[J]. 计算机工程, 2013(7):165-168.

[11] 孙知信. 网络异常流量识别与监控技术研究[M]. 清华大学出版社, 2010.

[12] 步山岳, 张海艳, 王汝传. 基于改进CUSUM算法的网络异常流量检测[J]. 计算机应用研究, 2009, 26(2).

[13] 李家春, 李之棠. 分布式入侵告警关联分析[J]. 计算机研究与发展, 2004, 41(11):1919-1923.

[14] 武斌, 杨义先, 郑康锋. 入侵检测中基于序列模式的告警关联分析[J]. 电子科技大学学报, 2009, 38(3):415-418.

[15] 张连华, 张洁, 白英彩. 基于数据挖掘的入侵检测告警关联分析研究[J]. 微型电脑应用, 2005(2):45-48.

[16] 郭洪波, 李炜, 王晶. 基于数据挖掘的增值业务网管告警关联分析模型[J]. 电信工程技术与标准化, 2011, 24(11):79-83.

[18] 尹群, 王丽珍, 田启明. 一种基于概率的加权关联规则挖掘算法[J]. 计算机应用, 2005, 25(4):805-807.

[19] 冯庆堂, 沈林成, 常文森. 基于概率数据关联的地形辅助导航算法[J]. 宇航学报, 2003, 24(5):439-443.

[20] J. A. Hartigan, M. A. Wong. A K-means clustering algorithm.[J]. Applied Statistics, 2013, 28(1):100-108.

[21] Arthur, David, Vassilvitskii, Sergei. k-means++: the advantages of careful seeding[C]// Eighteenth Acm-Siam Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2015:1027-1035.

[22] Kanungo T, Mount D M, Netanyahu N S, et al. An efficient k-means clustering algorithm: analysis and implementation[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2002, 24(7):881-892.

[23] Li H, Wang Y, Zhang D, et al. Pfp: parallel fp-growth for query recommendation[C]// ACM Conference on Recommender Systems. ACM, 2008:107-114.

[24] Wang K, Tang L, Han J, et al. Top Down FP-Growth for Association Rule Mining[C]// Advances in Knowledge Discovery and Data Mining, Pacific-Asia Conference, PAKDD 2002, Taipei, Taiwan, May 6-8, 2002, Proceedings. 2002:334-340.

# 附录 Pyspark聚类分析、关联分析代码

代码的链接如下：

1. K-means中将字符串替换成数值

<https://github.com/SuperSuperSuperSuper5/KDD-CUP-99-data-processing>

1. K-means中将Spark输出结果合并

<https://github.com/SuperSuperSuperSuper5/KDD-CUp-99-data-ijustwantresult>

1. K-means统计结果得出结论

<https://github.com/SuperSuperSuperSuper5/KDD-CUp-99-data-count>

1. DRAPA数据集的处理和输出

<https://github.com/SuperSuperSuperSuper5/How-to-process-LLDOS_1_inside>