一种分布式的僵尸网络实时检测算法

陈连栋[1] 张 蕾[1] 曲 武[2,3] 孔 明[1]
(国网河北省电力研究院 石家庄 050021)[1] (清华大学计算机科学与技术系 北京 100084)[2]
(启明星辰信息安全技术有限公司核心技术研究院 北京 100193)[3]

摘 要 僵尸网络通过控制的主机实现多类恶意行为,使得当前的检测方法失效,其中窃取敏感数据已经成为主流。鉴于僵尸网络实现的恶意行为,检测和减轻方法的研究已经势在必行。提出了一种新颖的分布式实时僵尸网络检测方法,该方法通过将Netflow组织成主机Netflow图谱和主机关系链,并提取隐含的C&C通信特征来检测僵尸网络。同时,基于Spark Streaming分布式实时流处理引擎,使用该算法实现了BotScanner分布式检测系统。为了验证该系统的有效性,采用5个主流的僵尸网络家族进行训练,并分别使用模拟网络流量和真实网络流量进行测试。实验结果表明,在无需深度包解析的情况下,BotScanner分布式检测系统能够实时检测指定的僵尸网络,并获得了较高的检测率和较低的误报率。而且,在真实的网络环境中,BotScanner分布式检测系统能够进行实时检测,加速比接近线性,验证了Spark Streaming引擎在分布式流处理方面的优势,以及用于僵尸网络检测方面的可行性。

关键词 大数据,僵尸网络,实时检测,Spark流计算
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.3.026

Distributed Real-time Botnet Detection Algorithm

CHEN Lian-dong[1] ZHANG Lei[1] QU Wu[2,3] KONG Ming[1]
(Information & Telecommunication Branch, State Grid Hebei Electric Power Company, Shijiazhuang 050021, China)[1]
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)[2]
(Core Research Institute, Beijing Venustech Cybervision Co. Ltd., Beijing 100193, China)[3]

Abstract Compared with other types of malware, botnets have recently been adopted by hackers for their resiliency against take-down efforts. Besides being harder to take down, modern botnets tend to be stealthier in the way they perform malicious activities by using the infected computer, making current detection approaches ineffective. Given the malicious activities botnets can realize, detection and mitigation of botnet threats are imperative. In this paper, we presented a novel approach for botnet detection, called distributed real-time botnet detection algorithm. It uses Spark engine, where Netflow related data are correlated as the host Netflow graph structure and the host access chain structure, and a feature extraction method based on the Spark Streaming is leveraged for exacting implicit characteristics. Meanwhile, this paper established distributed BotScanner detection system based on the Spark Streaming, which is a distributed real-time steam processing engine. We trained BotScanner system on the five representative bot families and evaluated BotScanner on simulated network traffic and real-world network traffic. The experimental results show that the BotScanner is able to detect bots in network traffic without the need of deep packet inspection, and achieves high detection rates with very few false positives. When the traffic data from the Internet service provider are very large, the BotScanner is able to detect botnets in real-time by adding the compute nodes, and BotScanner has approximate linear speedup. It proves the feasibility of Applying Spark Streaming engine to distributed botnet detection.

Keywords Big data, Botnet, Real-time detection, Spark streaming

1 引言

僵尸网络(Botnet)[1]是攻击者出于恶意目的,传播受控僵尸程序来控制大量主机,通过一对多的命令与控制信道(C&C)组成的网络,例如具有IRC协议、P2P协议、HTTP协议的僵尸网络。僵尸网络是在传统恶意代码形态(例如计算机病毒、网络蠕虫、特洛伊木马和后门工具)的基础上进行进化,并通过相互融合发展而成的目前最为复杂的攻击方式之一。当前,僵尸网络已经成为互联网最大的安全威胁之一,来自它们的攻击时常发生,并且在全世界互联网内蔓延,其攻击种类多样,例如分布式拒绝服务攻击、端口扫描、发送垃圾邮件、植入广告、网络钓鱼、非法利用用户主机资源等。由于巨

到稿日期:2015-02-01 返修日期:2015-04-20 本文受国家自然科学基金(60875029)资助。
陈连栋(1987—),男,硕士,高级工程师,主要研究方向为网络管理、信息工程、数据挖掘、网络与信息安全;张 蕾(1964—),女,硕士,高级工程师,主要研究方向为网络管理、信息工程、网络与信息安全;曲 武(1981—),男,博士后,主要研究方向为网络安全、大数据、数据挖掘,E-mail:quwu_ustb@163.com;孔 明 男,主要研究方向为网络安全。

大的经济利益，僵尸网络技术的优化和变种发展更为迅速，这也导致僵尸网络的检测和防卫更为困难。无论是现在还是将来，僵尸网络的研究都是网络安全领域重要的研究方向。

基于C&C流量实时检测僵尸网络通常是比较困难的，原因如下：(1)僵尸网络使用正常协议进行通信，与合法流量相似；(2)僵尸网络流量规模较低，持续时间较短；(3)在监控网络中，可能仅存在少量受控主机；(4)网络中可能包含加密或混淆的通信流量；(5)分析算法难以满足高速网络流量实时检测的要求。在本文中，通过对沙箱中的僵尸网络样本进行监控，同一家族的僵尸网络样本在流量上往往具有时空相似性和行为相似性，这些接近不变的特征有助于通过流量识别僵尸网络家族，进而识别僵尸网络。例如，僵尸程序Conficker家族网络具有通信时间间隔为632s或1057s、目的IP地址到源IP地址传输43kB、平均间隔为670s或1900s、上下行流量比为2.977∶1、每个流的持续时间为130ms或12s、FFT为0.011Hz或0.038Hz、IP地址熵值为8.72(该僵尸程序正在发动DDOS攻击)等特征。这些行为特征与正常流量差异较大，以此可以识别Conficker僵尸网络家族。

本文主要做了以下贡献：

(1)通过对5类不同僵尸恶意软件家族C&C通信数据的分析，提出利用Netflow数据实时检测感染僵尸恶意程序的主机。本文提出的检测系统是基于行为的，不是基于特征标签的，故能够处理加密和混淆通信。

(2)本文展示了一个基于学习的分布式僵尸网络实时检测系统BotScanner，其可以根据僵尸家族样本的实时流量自动生成检测模型，模型建立过程是在可控的沙箱或虚拟机环境下进行的。

(3)在BotScanner训练和检测过程中，基于Spark Streaming框架分布式构建了主机访问关系链和主机Netflow图谱两种流拓扑结构，并从中提取了8个特征，分别为关系链、平均间隔、平均流持续时长、平均流大小、流起始时间的傅里叶变换、IP地址熵、上下行流量比、流量负载熵。BotScanner系统基于这8个特征进行建模和检测。

(4)BotScanner是基于Spark分布式批处理平台建立的训练方法和检测模型，故BotScanner可以根据网络负载情况通过添加计算节点来提高性能。同时，BotScanner使用Hadoop的HDFS组件存储数据，也可以横向扩展。

(5)基于Spark平台，开发了BotScanner分布式检测原型系统。理论分析和实验结果表明，通过调整不同的行为特征，BotScanner既可以应用在千兆流量的企业网，又可以应用到运营商网络中。实验中，基于真实流量数据的测试表明，BotScanner能够获得较高的检测率和较低的误报率。此外，BotScanner系统的可扩展性、实时性、检测性能总体上优于当前主流的僵尸网络检测系统。

本文第2节首先对僵尸网络的结构及相关检测技术进行介绍，然后对分布式与并行计算方法和架构进行简要描述；第3节详细介绍BotScanner分布式检测系统的实现细节；第4节详尽介绍BotScanner分布式检测系统的训练和实验评估；最后进行总结和展望。

2 背景和相关工作

2.1 僵尸网络和检测方法

目前，僵尸网络已经成为互联网最大的安全威胁之一。同时，僵尸网络由于本身的特性，已经成为进行敏感信息偷窃、垃圾邮件发送、DDoS等攻击的高效手段。为了应对僵尸网络的攻击手段，安全研究人员已经在僵尸网络检测领域展开了深入的研究，下面讨论近年来开展的相关研究。

在僵尸主机检测领域，基于网络的僵尸主机检测主要分为两个研究方向：垂直关联检测和水平关联检测。在垂直关联检测中，为了评估主机受恶意代码感染的情况，通常使用主机流量分析来检测扫描、垃圾邮件和C&C通信等恶意行为。例如，BotHunter系统[2]利用签名检测和异常检测组件对一个典型的感染生命周期进行检测。Goebel等人[3]和Binkley等人[4]分别基于IRC的网络流量中僵尸主机频繁使用的昵称模式进行建模和检测。然而，这些技术仅仅适用于特定的僵尸网络结构，或者依赖于出现一种特定的僵尸恶意程序感染生命周期。而且，大多数检测技术依赖于噪声行为，例如扫描、垃圾邮件或是DDoS攻击流量。Wurzinger等人[5]和Perdisci等人[6]分别通过自动分析受感染主机的行为，生成检测签名。这些方法的核心思想是，受控主机收到C&C服务器的命令，然后以一种特定的方式进行回复，该过程能够提取签名。这些方法尽管都能够获得较高的检测率和有限的误报率，但其局限性在于它们都要求深度包解析，因此对于加密和混淆的C&C通信无能为力。Giroire等人[7]展现了一种方法，该方法的主要研究对象是临时访问关系，他们还提出了目标原子和持久性的概念。目标原子之间使用一种通用的服务或Web地址进行通信，构成了目标原子组；而持久性是对目标原子组的临时规律性进行多重粒度的评估。其核心思想是使用每台主机一段时间的初始化连接进行训练，然后将结果分组到目标原子组。随后，一段时间内固定一致的目标原子组构成了白名单，与白名单相比出现较少的连接将被记为异常，以此识别一个C&C服务器。第二个研究方向就是来自多台主机的网络事件水平关联，这些事件通常被包含在相似的、恶意的通信中。BotSniffer[8]、BotMiner[9]、TAMD[10]系统和Strayer等人[11]的工作都是围绕此展开的。除了Strayer等人的工作是针对IRC分析的，其他3个系统的检测原理都与僵尸网络结构无关，故对于IRC、HTTP和P2P类型的僵尸网络都是有效的。然而，不同主机之间的关联行为要求至少在同一个监控网络中存在两台受控主机，且感染同一种僵尸恶意代码。因此，这些技术对于单个受控主机的检测无能为力。此外，这些检测技术要求能够观测到一些噪音行为，即根据噪音进行检测。但是，与过去几年相比，低速、低噪音和利益驱动的僵尸网络行为正在增多。另外一类检测P2P僵尸网络的方式是以BotGrep系统[12]、BotTrack系统[13]、BotFinder系统[14]、Coskun等人[15]提出的FoE方法、BotSuer检测系统[16]、Yang等人提出的Botnet在线监控和离线检测方法[17]、Amini等人提出的基于非监督学习的Botnet检测方法[18]、Garg等人提出的基于网络行为的Botnet检测方法[19]为代表。这类方法利用僵尸网络潜在的通信架构进行检测。BotGrep利用指定的分布式哈希表交互行为进行检测，BotTrack和BotFinder两个系统都是利用NetFlows数据进行通信异常检测，但是统计方法不同。对于FoE方法，无需进行包深度解析，仅需要进行交互式通信图谱计算进行检测。然而，除了BotFinder系统，其他系统都是利用蜜罐捕获僵尸恶意样本，然后通过监控僵尸恶意程序之间的通信定位网络中其他的受控主机，即这些方法要求蜜罐中存在一个活跃的僵

尸源，这在实际使用中往往受限。BotFinder 方法同样需要利用蜜罐捕获僵尸恶意样本，但该方法使用沙箱或虚拟机来执行，为其运行行为和通信行为建模，最后检测。在文献[20]中，Vania 等人对 Botnet 检测的方法和发展趋势进行了综述，并提出了独特的见解。以上这些方案中提到的方法为本文提出的 BotScanner 系统提供非常好的借鉴思路，但这些方法对于高速大流量实时数据流的检测略显不足，这导致以上系统的实际部署应用受限。

在恶意代码分布式检测领域，研究成果不多，且多使用离线分布式统计和在线检测。Jerome 等人利用分布式 PageRank算法检测僵尸网络[13]，其算法是基于 Hadoop 平台的 Map-Reduce，通过分布式迭代计算每个主机的 PageRank 评分进行检测。Zhao 等人[21]主要针对发送垃圾邮件的僵尸网络进行研究，其方法是利用邮件之间的发送关系构造关系网络，然后通过计算子图来定位垃圾邮件，并形成黑名单列表用于检测。Jiang 等人[22]利用僵尸网络 C&C 流量中存在的关联关系进行检测，其检测过程是在线包解析提取特征，然后基于 Hadoop 平台离线提取关联关系，并将异常的多级关联标记为受控僵尸主机。这 3 种方法由于采用了 Hadoop 平台提供的 Map-Reduce 框架，其本质上都是一种离线批处理方法，统计结果可以用于在线检测。但其在线检测部分依然是串行架构，对于高速的网络流量难以实时处理。针对现有方法存在的局限性，本文提出一种基于 Spark 的僵尸网络分布式实时检测算法，该算法通过对僵尸恶意软件的 C&C 通信流量进行建模，然后通过提取 Netflow 内在的拓扑特征检测僵尸网络。本文采用了 Spark Streaming 分布式流处理技术，所以可以通过水平扩展方式满足运营商网络环境的高速流量实时分布式处理要求。

2.2 分布式与并行计算

分布式和并行计算领域已经被研究多年，存在许多优秀的研究成果。大规模并行处理技术，即 MPP(Massive parallel processing)[23]，是多个处理器处理同一程序的不同部分时该程序的协调过程，工作的各处理器运用自身的操作系统和内存。MPI(Message Passing Interface)技术[24]和 PVM(Parallel Virtual Machine)技术[25]开发了相应的软件库来支持并行计算。分布式数据库是并行数据处理的另外一个大分类。

新出现的云计算模型，例如 Map-Reduce[27]、Hadoop、Spark[26]，可以使我们在集群上使用更简单的程序实现更有效的大规模静态数据和高速实时数据分析。以上这些技术采用的是分阶段计算概念，即使用调度策略、负载平衡和自动失败恢复策略进行计算。这些技术为重新思考网络安全提供了一个机会，网络安全的核心工作就是处理大量的日志和流量数据，本文的工作是本方向最早的尝试之一。Spark 是 UC Berkeley AMP lab 开发的开源的类 Hadoop Map-Reduce 的通用并行计算框架，Spark 基于 Map-Redcue 算法实现的分布式计算拥有 Hadoop Map-Reduce 所具有的优点。但不同于 Map-Reduce 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 更适用于数据挖掘与机器学习等需要迭代的 Map-Reduce 算法。Spark Streaming 是构建在 Spark 上处理流数据的框架，基本的原理是将流数据分成小的时间片断(几秒)，以类似批量处理的方式来处理这小部分数据。Spark Streaming 构建在 Spark 上，一方面是因为 Spark 的低延迟(100ms+)执行引擎可以用于实时计算，另一方面相比基于 Record 的其它处理框架(如 Storm)，RDD 数据集更容易做高效的容错处理。此外小批量处理的方式使得它可以同时兼容批量和实时数据处理的逻辑和算法，适用于一些需要历史数据和实时数据联合分析的特定应用场合。

通过对以上技术的分析，我们选择 Spark Streaming 分布式流处理平台实现 BotScanner 系统。在技术选型过程中，曾经考虑过使用更为主流的 Hadoop Map-Reduce 技术，但其批处理的工作方式只适用于离线数据处理，不能满足实时性的要求。因此，为了满足实时性、分布式和吞吐量的需求，选择 Spark Streaming 平台实现 BotScanner 检测系统。基于 Spark Streaming 的 Netflow 处理过程的基本原理是将输入的 Netflow 数据流以时间片为单位进行拆分，然后以类似批处理的方式处理每个时间片数据。首先，Spark Streaming 把实时输入数据流以时间片 Δt(如 1s)为单位切分成块；然后把每块数据作为一个 RDD，并使用 RDD 操作处理每一小块数据；每个块都会生成一个 Spark Job 处理，最终结果也返回多个块。

3 BotScanner 系统

通过比较当前流量的 Netflow 统计特征与僵尸网络历史流量 Netflow 行为特征，BotScanner 可以检测主机感染的恶意代码。在此过程中，BotScanner 包含两个阶段，即训练阶段和检测阶段。在训练过程中，基于机器学习算法，BotScanner 学习不同的僵尸网络家族的 C&C 通信特征。然后，BotScanner 使用这些特征建立检测模型来识别相似的流量。在检测过程中，这些模型被用来检测网络流量。由于训练过程中充分考虑了加密 C&C 通信问题，即使僵尸程序使用了加密的 C&C 通信，BotScanner 也能够识别潜在的僵尸程序感染。

3.1 系统架构

图 1 描述了 BotScanner 系统的训练阶段和检测阶段所包含的步骤：第一步，BotScanner 需要获取 Netflow 作为输入。在训练阶段，BotScanner 通过在沙箱或虚拟机中执行僵尸程序样本，捕获这些样本产生的流量，并提取 Netflow 特征(或利用路由器、交换机上的 Netflow 特征)。第二步，BotScanner 构造主机关系链和主机 Netflow 图谱。第三步，通过主机关系链构造可信主机关系链特征。此外，根据主机 Netflow 图谱，提取 7 个统计特征，分别为主机 Netflow 图谱中的两个流之间的平均间隔、平均流持续时长、平均流大小、对于流起始时间的傅里叶变换、IP 地址熵、上下行流量比、流量负载熵。最后，BotScanner 利用前面提到的 8 个特征建立模型。在模型建立过程中，BotScanner 使用聚类算法处理观测到的特征值集合。由于需要检测多类僵尸网络家族，而且各类僵尸网络家族特征相关性难以判别，因此对每个特征将分别进行建模。例如，一个僵尸网络家族可能在 C&C 通信上表现出相似的周期性，但是每个连接却传输了不同大小的流负载。多个特征聚类的联合将形成 BotScanner 的最终僵尸网络检测模型。而且，在检测过程中，可以辅以黑白名单列表进一步提高系统的检测精度。

在检测阶段，BotScanner 可以接受不同粒度的数据，这些数据既可以来自于标准的 Netflow 设备，例如交换机、路由器等，也可以来自网络流量，用户可进行自定义的 Netflow 特征提取。不同的粒度对应不同的应用场景，也对应着轻微差异

的检测精度和误报率。使用自定义 Netflow 特征的 BotScanner 多部署在流量相对不大的企业网出口，这是由于自定义 Netflow 特征提取时间代价较高，流量过大将造成丢包严重，导致自定义 Netflow 特征不精确。BotScanner 使用训练阶段建立的模型进行僵尸网络检测。而使用标准的 Netflow 特征的 BotScanner 多部署在运营商网络。值得注意的是，BotScanner 可选输入数据粒度的特点，使得 BotScanner 不依赖于网络流量的负载信息，却可以产生接近的检测精度和误报率。在本文中，将不进行二者的检测精度比较，仅将其作为部署说明使用。

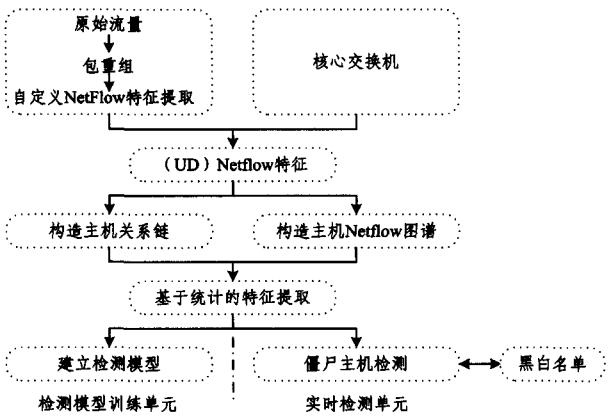


图 1　BotScanner 系统架构

在训练阶段，僵尸恶意样本在沙箱或虚拟机这类可控环境下执行，所有的网络流量都将被捕获和记录。该阶段需要正确地分类僵尸恶意样本，即同一家族的不同样本需要统一分析。在僵尸样本分类过程中，BotScanner 将动态检测和静态检测相结合，选用业界知名的检测和标注系统 Anubis 与 VirusTotal，分别用于动态检测和静态检测。当然，不正确的分类肯定会存在，这可能会影响生成检测模型的质量。但后期的实验结果表明，BotScanner 对于训练数据集中的噪音具有一定程度的容错性。

3.2　Netflow 生成

要生成主机访问关系链和主机 Netflow 图谱，首先需要获得 Netflow 数据。NetFlow 是一种数据交换方式，通常定义为在一个源 IP 地址和目的 IP 地址间传输的单向数据包流，且所有数据包具有共同的传输层源、目的端口号。Netflow 数据主要有两个来源，分别为网络设备吐出和自生成。交换机和高端路由器吐出的 NetFlow 数据记录由过时的数据流及详细的流量统计数据组成。这些数据流中包含来源和目的相关信息，以及端到端会话使用的协议和端口。对于自生成的 Netflow 数据，BotScanner 根据 IP 地址、端口号和协议可以确定一条流，即具有相同的五元组信息（源 IP，目标 IP，源端口，目标端口，传输层协议）的 IP 报文重组成的一条流。对于每个连接，除了五元组信息外，BotScanner 同时也会提取连接的起始时间、结束时间、传输的字节数、数据包数、上下行流量比、流量负载熵等特征信息。

3.3　检测特征提取

3.3.1　主机访问关系链

集中式僵尸网络主要以具有 IRC 协议和 HTTP 协议的僵尸网络为主，为防止单点故障通常采用多个 C&C 服务器，同时多个 C&C 服务器的 IP 地址对应一个域名。受控主机通常使用心跳技术与这些 C&C 服务器保持连接。分布式僵尸网络主要以基于 P2P 协议的 C&C 为主，没有单点的 C&C 服务器问题，通常采用发布和订阅方式进行消息通信。僵尸网络控制者发送控制或者命令，受控主机以一定模式向其相邻节点发送心跳信息。此外，受控主机也通常以一定模式连接相邻节点以交换心跳消息。在分布式僵尸网络中，每个受控主机都维护一个相邻节点列表，并以一定模式与名单中节点进行通信，即不断与同一组节点进行通信。上述僵尸网络模型中，每个受控主机维护一个相邻节点列表，以一定模式访问名单中的僵尸节点以获取控制和命令信息。大部分僵尸网络符合上述模型，主机关系链主要描述此类僵尸网络。

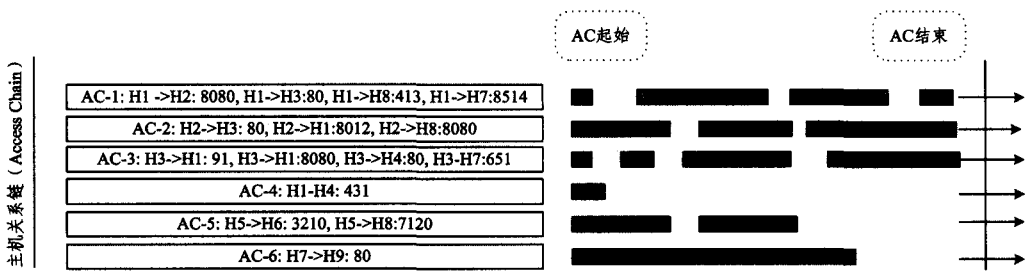


图 2　主机关系链

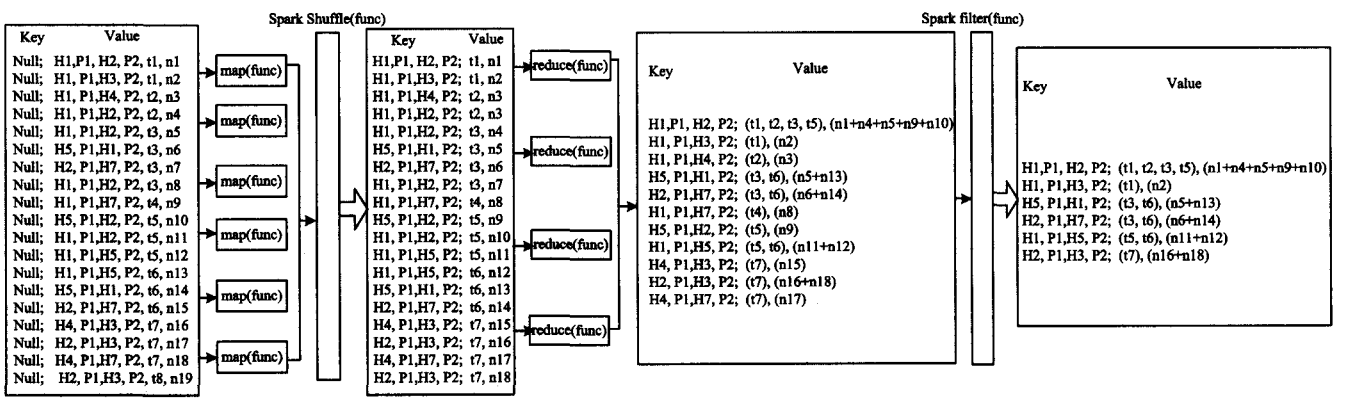


图 3　基于 Spark Streaming 的主机关系链生成示意图

受控主机连接其邻居名单中前后相邻节点的流总是“相继出现”，即一个流先出现，另一个流紧跟前一个出现，这些流具有前驱后继的特征，BotScanner 系统中称之为“主机关系链”，如图 2 所示。根据关系链中流的个数不同，将其分为二级和多级关系链。二级关系链指前后两个流的前驱后继关系，多级关系链指前后多个流的前驱后继关系。相比之下，合法用户的网络行为较为随机，几乎不会呈现明显的关系链。因此，BotScanner 系统通过提取流间的关系链识别 C&C 流量，从而检测出僵尸主机。基于 Spark Streaming 的主机关系链提取示意图如图 3 所示，包括 Map(func)、Shuffle(func)、Reduce(func)、Filter(func)等过程。其中，H、P、t 和 n 分别表示主机、端口、获取该数据流的时间窗标识和该数据流出现的次数。例如，($H1$，$P1$，$H2$，$P2$，$t1$，$n1$)表示在时间窗口 $t1$，数据流从源主机 $H1$ 源端口 $P1$ 到目标主机 $H2$ 目标端口 $P2$ 出现了 $n1$ 次。生成的主机关系链，例如[($H5$，$P1$)，[($H1$，$P2$，$n5$)，($H2$，$P2$，$n9$)]]，表示主机($H5$，$P1$)具有三级关系链特征。

3.3.2 主机 Netflow 图谱

主机(UD)Netflow 图谱和主机关系链是 BotScanner 系统最为核心的两个概念，主机关系链在上文已经进行了描述。BotScanner 系统中，Netflow 图谱是两个网络节点间(主机 & 端口)时序的流序列，图 4 显示了不同形状的 Netflow 图谱。例如 *NG*-4 使用 431 端口，从主机 $H1$ 到 $H4$ 显示了高度的规律行为。这种规律性使得 BotScanner 能够提取 Netflow 图谱的统计特征。在 Netflow 图谱 *NG*-4 中，接近常量的流之间时间间隔和持续时间使得这两个特征能够精确描述整个 *NG*-4 图谱。为了获得更有意义的统计数据，BotScanner 需要 Netflow 图谱包含一定量的连接参数，即 $|NG|_{min}$，在 BotScanner 原型中，$|IT|_{min}$(30≤$|IT|_{min}$≤70)表示 Netflow 图谱中包含的最小连接数。Netflow 图谱的最小连接数与僵尸网络实际通信是一致的，即 C&C 网络通信通常包含多个主机与 C&C 服务器之间的连接。在自动分析僵尸恶意样本过程中，最主要的挑战是如何区分恶意 C&C 通信与正常 C&C 通信，在训练过程中，正常 C&C 通信为噪音数据。基于 Spark Streaming 的主机 Netflow 图谱生成示意图如图 5 所示，包括 Map(func)、Shuffle(func)、Reduce(func)、Filter(func)等过程。其中，H、P、t 和 n 分别表示主机、端口、获取该数据流的时间窗标识和该数据流出现的次数。例如，($H1$，$P1$，$H2$，$P2$，$t1$，$n1$)表示在时间窗口 $t1$，数据流从源主机 $H1$ 源端口 $P1$ 到目标主机 $H2$ 目标端口 $P2$ 出现了 $n1$ 次。生成的主机 Netflow 图谱，例如[($H1$，$P1$，$H2$，$P2$)，[($t1$，$t2$，$t3$，$t5$)，($n1$，$n4$，$n5$，$n9$，$n10$)]]，表示主机($H1$，$P1$)到主机($H2$，$P2$)之间的 Netflow 图谱。事实上，为了检测网络可达性、同步时间或为发送垃圾邮件，一些僵尸变种甚至能够故意伪造访问合法站点的良性流量，以此隐匿自己的 C&C 通信。BotScanner 使用两种方式过滤无关流量并识别相关流量：(1)使用网络访问白名单，例如 Microsoft Update、Baidu 服务等；(2)可以利用第三方的知识库或使用已知的静态特征、通信模式与训练流量进行比较，也可以将目标 IP 地址与已知的僵尸网络 C&C 服务器进行比较。值得注意的是，若与僵尸网络的 C&C 通信无关的流量被包含到模型建立过程中，则表明预测模型的可信度较低。但是，实验结果表明，可信度较低的模型对于检测结果影响不大。

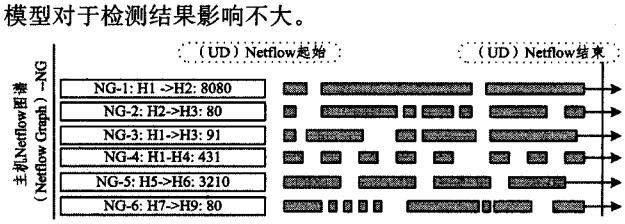


图 4 主机 Netflow 图谱

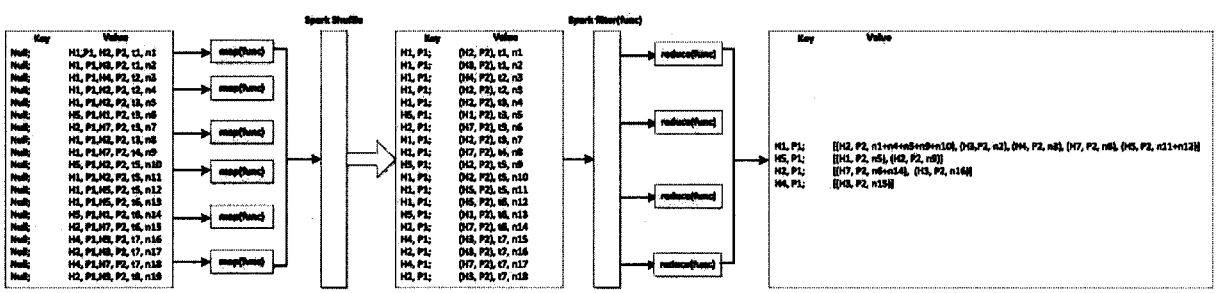


图 5 基于 Spark Streaming 的主机 Netflow 图谱生成示意图

3.3.3 检测特征提取

主机 Netflow 关系链构建之后，BotScanner 提取主机关系链特征，即是否具有二级关系链和多级特征，该特征较为简单，上文已经描述完毕。对于主机 Netflow 图谱，为了分类图谱中的子序列，BotScanner 处理每个图谱抽取统计特征。BotScanner 主要聚焦于以下 7 个统计特征，分别为主机(UD) Netflow 图谱中两个流之间的平均间隔、平均流持续时长、平均流大小、流起始时间的傅里叶变换、IP 地址熵、上下行流量比、流量负载熵，具体描述如下。

(1)平均间隔：在主机 Netflow 图谱中，两个流子序列之间起始的时间间隔。僵尸主控节点必须保证其所控节点能够接收到新的控制与命令。通常，从 C&C 服务器到受控僵尸主机之间的通信采用 Push 模式是不可能的。其原因是，许多受控僵尸主机是在私有网络中，并处于网络地址转换设备 NAT 后面，或者是还没有注册到 C&C 服务器上。我们假设大部分受控僵尸主机与 C&C 服务器之间的通信都使用一个常量的时间间隔(或者一个固定范围内的随机值)，这将致使通信中可检测的周期性。对于这类通信模式，僵尸主控节点必须平衡僵尸网络的可扩展性、灵活性和 C&C 服务器连接被检测到的风险。正如前面所提，一些僵尸网络变种为规避特征提取和恶意代码检测系统，开始采用随机和合法连接。其他方法，例如每天定时连接也遭遇到主机时钟不同步的问题。而且，僵尸恶意代码作者可以通过精心设计恶意代码使其不显示周期行为。但是，模仿随机和合法通信是相当困难的，而且也是可检测的。基于对于不同的僵尸恶意代码家族的观察，当前大部分僵尸恶意代码都符合上面的假设，同时表

现出松散的周期 C&C 通信特征。

(2)平均持续时长:在主机 Netflow 图谱中,每个流子序列持续时间的均值。通常在受控僵尸主机没有收到新命令的情况下,大部分通信仅包含简单的握手信息,即受控僵尸主机请求"新指令",C&C 服务器返回"没有新指令"。因此,在主机 Netflow 图谱中,可以看出该过程的持续时间是相近的。

(3)平均流大小:在主机 Netflow 图谱中,平均流大小包含两个子特征,分别为平均源字节数和平均目标字节数。通过使用源和目标字节数切分两个方向的通信,我们能够将请求信道与命令传输区分开。也就是说,请求更新信息通常具有相同的大小,但来自 C&C 服务器的真实数据是可变的。因此,主机 Netflow 图谱可能包含许多具有相同源字节数的流子序列。相似的判断也可应用到目标字节数,例如,来自 C&C 服务器的响应具有固定的格式。

(4)流起始时间的快速傅里叶变换[14]:为检测潜在的 C&C 通信规律,使用快速傅里叶变换算法(Fast Fourier Transform algorithm,FFT)处理 C&C 通信的二进制抽样。在此过程中,我们对主机 Netflow 图谱进行抽样,每个子序列起始位置设定为 1,子序列中间位置则设定为 0。通过这种方式,一个主机 Netflow 图谱被映射为一个二进制序列。为计算高质量的 FFT,使用 Netflow 图谱中最小时间间隔的 1/4 作为抽样间隔,避免欠采样问题。尽管如此,若两个流子序列之间距离非常小,其他流子序列之间距离非常大,该抽样方法将会产生大量的数据点。在此情况下,需要限制单个 Netflow 图谱中包含的采样点数为 65536,即 2^{16},同时接受轻微欠采样情况。在使用该值的情况下,FFT 是最快的,而且实验中仅有少量数据点被欠采样成单一的点。更为精确地描述如下,即对于观测到的 C&C 图谱,12%显示欠采样,但是仅平均 1%的起始时间被抽样到了一个采样点。在下一步,为抽取最重要的频率特征,需要计算主机 Netflow 图谱 FFT 的功率谱密度。FFT 的峰值对应着时间周期性,并且对于主机 Netflow 图谱中大跨度的间隔具有耐受性。在实际测试环境中,僵尸网络恶意代码的 C&C 服务器通信是周期性的,然后会停滞一段时间。在一定的时间窗口内,若恶意代码作者设计可随机变化的 C&C 通信频率的僵尸代码,这种随机变化将会降低 FFT 的峰值。然而,FFT 的峰值仍然可检测并保持同样的频率,因此可以通过 FFT 的峰值检测僵尸网络通信。

(5)IP 地址熵:同一网络的一定范围内,计算正常流量中(UD)Netflow 五元组的 IP 地址熵,熵值都大致相同。如果出现异常,则异常流量将改变(UD)Netflow 的 IP 地址熵,熵值也将与正常情况下的熵值出现很大差别。不同类型、不同比例的异常流量,熵值也有很大不同。例如,在僵尸网络感染期间,受控的主机会在很短时间内连接到其他许多主机。受控主机建立的开启连接会占多数,熵随之减小。同样,数据流中的目标 IP 地址会比正常流量中的 IP 地址随机得多。也就是说,目的 IP 地址的分布会更分散,导致信息熵较高。在僵尸网络发动 DDoS 攻击阶段,目的 IP 地址熵值较小,源 IP 地址熵值较大;网络扫描探测阶段目的 IP 地址熵值较大,源 IP 地址熵值较小。由于正常网络流量具有较为稳定的 IP 地址熵,异常流量将破坏这种稳定,从而可以从 IP 地址熵的角度进行僵尸主机检测。

(6)上下行流量比:用户正常访问网络时产生的上行流量比下行流量小。流量异常是指用户访问网络时产生的上行流量比下行流量大得多。主要有两种原因造成流量异常:1)使用 P2P 软件(电驴、迅雷、PPLive、UUSEE 等)在外网下载、看视频(如电影、电视)会产生大量的上行流量,为其他使用 P2P 软件的用户提供自己计算机上的数据;2)感染僵尸病毒,大量向外网发送恶意代码自动收集危害计算机安全的敏感数据。对于正常的 P2P 软件,可以使用白名单列表方式进行过滤,剩下的为僵尸软件或恶意代码造成的可疑流量。

(7)流量负载熵:近几年来,新的僵尸程序将加密和混淆技术用于其 C&C 通道,以避免 IDS、防火墙或其他方式的网络侦听。由于流量负载部分完全是加密或混淆的,基于特征的检测方法难以提取加密通道的特征,因此检测僵尸网络难以实现。然而,由于加密算法导致负载中字符出现的随机性大大提高,每个字符出现的概率变小。也就是说,信道流量加密后,其熵值比较高。因此,可以通过熵值判断信道是否加密。然后,结合白名单列表方式过滤掉合法加密流量,剩下的流量为僵尸软件或恶意代码造成的可疑流量。

3.4 建立检测模型

通过聚类主机关系链包含的 1 个特征和主机 Netflow 图谱包含的 7 个特征分别为关系链、平均间隔、平均流持续时长、平均流大小、流起始时间的傅里叶变换 FFT、IP 地址熵、上下行流量比、流量负载熵。通过僵尸恶意代码行为进行观测,这 8 个特征在总体上是无关的,因此对每个特征分别进行聚类。例如,一个僵尸代码的两个版本可能连接不同版本的 C&C 服务器,传输不同大小的流,IP 地址熵不同,流量负载熵不同等,然而,这两个版本的僵尸代码通信的周期模式仍然相同。聚类过程之后,可以明显观测到相当大的聚类簇含可疑的和真实的僵尸恶意代码特定的行为。此外,一些小的聚类簇包含多样的数据,这样的聚类簇往往具有较低的聚类质量,甚至对应的主机访问关系链和 Netflow 图谱也相当独特。训练结束后,最终的模型将包括 7 个聚类簇集合,每个集合对应一个特征,集合内包含了该特征的期望值。

为了聚类主机关系链和 Netflow 图谱的特征,BotScanner 使用 Spark 的机器学习库 MLLib 中的分布式 Kmeans ‖ 算法[28],该算法解决了人为确定初始聚类中心时不同的初始聚类中心可能导致完全不同的聚类结果的问题。而且,由于采用 Spark 的分布式算子,Kmeans ‖ 算法能够处理海量数据。实验结果表明,Kmeans ‖ 算法能够较好地适应本文的应用场景。

在计算完聚类中心和成员后,使用聚类质量评估函数判断各个聚类簇的质量。聚类评估函数定义为 $q_{du}=\exp(-\beta\cdot\frac{sd}{c})$。其中,$sd$ 为标注方差,c 为均值,β 为控制因子,默认值为 2.71。一般情况下,大的聚类簇具有更高的内部相似性、更多样的聚类子簇。所有聚类簇 q_clu 的均值是评估特征向量相似性的指标。高的聚类质量表明,大多数的僵尸恶意代码生成高度相似的 Netflow,抽取的特征向量也相近。如果 Netflow 是多样的,则主机关系链和 Netflow 图谱也是多样的,就会产生更多低质量的聚类簇。然而,这并不是一个糟糕的情况。假定一个僵尸恶意代码家族样本,其 C&C 通信具有确定的周期间隔特征,但为了规避检测,增加了人为的随机流量。然后,从该样本生成的流量中提取相关的 C&C 通信信息。在聚类过程中,由大多数僵尸恶意代码家族样本产生的

确定时间间隔的 C&C 通信被聚类成高质量的聚类簇，然而随机的流量被聚类成低质量的、松散的聚类簇，且具有高的标准方差。尽管这些低质量的聚类簇降低了整个聚类的平均聚类质量，但是捕捉到真实 C&C 通信的聚类簇仍然是高质量的，而且表现了相关僵尸恶意代码的行为。

3.5 受控僵尸主机检测

为检测一个给予的特征向量 V 是否匹配训练好的模型 M，需要将 V 中的每个特征与模型中的聚类簇进行比较。例如，如果特征向量 V 中的平均时间间隔特征处于 M 的一个聚类簇中，检测算法将认定为一次命中，并调高检测得分值 γ_M。检测得分值 γ_M 的提升幅度主要依赖于聚类簇的质量和特征向量的质量(即主机关系链和主机 Netflow 图谱的质量)。这些质量定义在一定程度上反映了主机关系链和主机 Netflow 图谱建立以及特征提取过程的不确定性。此外，还需要考虑平均时间间隔特征处于何值时可以被认为具有周期性。通常，聚类质量越高，检测得分值 γ_M 提升越快。更确切地说，设 $\gamma_M=\gamma_M+q_{clu}\cdot\exp(-\beta\cdot\frac{sd}{c})$。其中，$\beta=2.71$。对于 γ_M，其范围被定义为 $[-2*sd_v, 2*sd_v]$，即对于所有的特征值，命中聚类簇的中心时，标注方差的 2 倍。对于 γ_M 范围的限制主要是从处理速度方面进行考虑，以优化处理性能。在数学上，对于指数评分函数的描述是递减速度很快。因此，与聚类中心相比，其他特征值对于评分的贡献范围为 0 到 γ_M 之间，多于两倍的标准方差。

对于不同的僵尸网络家族，为了能够命中多个模型，需要为每个模型维护一个 γ_M。需要注意，在训练过程中引入的人造训练数据(与 C&C 通信无关的流量)将会造成低质量的聚类簇，这仅仅会导致 γ_M 轻微上升。通过这种方式，BotScanner 系统隐含着对训练中引入的噪音的一定程度的免疫能力。最后，将最高的评分 γ 与 BotScanner 系统报警阈值 α 进行比较，若 $\gamma>\alpha$，则该模型判断为匹配，BotScanner 发出告警。为了降低误报率，BotScanner 系统不仅仅是依赖单一的特征值进行判别，用户可以根据实际环境情况设定最小的命中特征数 h。也就是说，除了每个特征都需要满足 $\gamma>\alpha$ 外，至少存在 h 个这样的特征向量，$1\leqslant h\leqslant 8$。这样的约束存在可以降低 BotScanner 系统由于突发的单一特征匹配导致的误报。例如，对于给定特征向量中的平均时间间隔和 FFT 两个特征，同时满足 $\gamma>\alpha$，若设定 $h=3$，BotScanner 系统需要额外一个特征满足 $\gamma>\alpha$ 才会告警，例如主机关系链、流量负载熵、IP 地址熵等。

4 BotScanner 系统的训练和实验评估

4.1 模型训练

本文使用 5 个不同的僵尸恶意代码家族训练 BotScanner 系统，这 5 个家族当前在互联网上非常活跃，并且非常有代表性。更确切讲，为了能够保证 BotScanner 系统所使用的训练样本是活跃的、相关的，我们观测了 Anubis 在 2013 年 5 月份的检测样本，并从中选出适合 BotScanner 系统训练的样本。Anubis 每天收到并分析成千上万的样本，在动态检测领域具有权威性。为了捕捉僵尸网络产生的通信流量，基于 Xen 构建了可控的 Windows XP 虚拟机环境。同时，该虚拟机与互联网相连，且包含真实的用户数据。在可控的虚拟机环境下，每个僵尸恶意代码家族平均测试了 32 个变种样本，并捕获了所有的网络流量。在实验中，我们对垃圾邮件和 DoS 攻击进行了限制。用来训练的 5 个僵尸恶意代码家族如下：

1) Conficker，该僵尸软件在业内也被称为 Downup、Downadup 或 Kido，目前全球已有超过 1500 万台电脑受其感染。Conficker 僵尸程序传播主要通过运行 Windows 系统的服务器服务的缓冲区漏洞。它使用特定的 RPC 请求在目标电脑上执行代码。

2) Banbra，一种特洛伊木马和间谍程序，用来进一步下载和安装恶意代码组件。

3) Black Energy，该僵尸软件起初被用来进行 DDoS 攻击、垃圾邮件发送和银行诈欺。Black Energy 僵尸软件最初使用一种基本的加密技术来使其可执行文件不被杀毒软件察觉，并使用 Base64 编码来扰乱其通信。最新版本的 Black Energy 使用了更强大的 RC4 流加密的一个变种来编码其通信。

4) Rustock，全球最大的僵尸网络之一，由僵尸程序 Rustock 感染的受控主机组成，其主要的非法行为是利用受控主机发送垃圾邮件。此外，受控的主机可能被窃取敏感信息，并被用来进行 DDos 攻击。

5) Pushdo，也称为 Pandex 或 Cutwail。PushDo 僵尸网络始于 2007 年 1 月，它是全球第二大的垃圾信息僵尸网络，未臭名远播的原因在于作者使用了许多不同的技术使它难以被侦测，然而 PushDo 不但主导全球大量的垃圾信息发送，同时也是犯罪集团用来散布恶意程序的主要管道。新的 Pushdo 变种通过假冒的 HTTP 请求，使得 Pushdo 命令和控制流量融入到合法流量，来攻击各种网站。

表 1 展示了训练过程中僵尸家族样本的详细分布和与之相关的主机关系链和主机 Netflow 图谱。在表 1 中，聚类质量反映了质量评分函数的结果。聚类质量分值越高，表明聚类簇中特征向量越接近，方差越低。同时，验证了我们的核心假设：同一僵尸恶意代码家族的不同变种将会产生类似的 C&C 通信流量，这类流量可以使用聚类算法进行有效的描述。而且，正如前面所说，较低的聚类质量并不意味着无效的恶意行为捕获。例如，在 Conficker 模型中，对于每个特征，最高的聚类簇质量高达 0.92。尽管具有较低聚类质量的小聚类簇降低了总体的聚类质量，但大的、高质量的聚类簇仍然能够对 Conficker 的通信行为给予较好的表述。对于 Pushdo，由于其样本通信行为的高可变性，生成的聚类簇也是多样的，导致建立的模型需要综合考虑多个特征。对于聚类方法，基于 Spark 的机器学习库 MLLib，本文引入文献[28]提出的 Kmeans ‖，对于每个僵尸网络家族，每个特征获得的平均聚类簇个数为 4.51。

表 1 用于训练的恶意代码家族描述

家族	样本	主机关系链	Netflow 图谱	主机关系链 & Netflow 图谱	聚类质量
Conficker	22	19	21	18	0.92
Banbra	31	21	28	17	0.76
Black Energy	28	16	61	12	0.74
Rustock	34	25	42	20	0.81
Pushdo	35	29	109	29	0.69
均值	30	22	52.2	19.2	0.784

4.2 系统实现和性能测试

为了兼容 Spark 的 Java API，我们实现 BotScanner 系统

时也使用 Java 语言。对于全包捕获和包重组环境，BotScanner 系统引入开源的 IDS：OSSEC(http://www.ossec.net/)。在训练和检测过程中，由于 BotScanner 系统采用了 Spark Streaming 大规模流式处理引擎，无论是对于千兆还是万兆设备流量都可以近实时处理。值得注意的是，本文使用的全包捕获和包重组过程是在 Spark Streaming 环境进行，即 Spark Streaming 的 Job 中将调用 OSSEC 的接口。在实际环境中，一个 10 个节点(4 路 8 核 3.2GHz CPU，128GB 内存)的集群被用来部署大数据环境，大数据平台采用 Cloudera 公司的 CDH。LabCapture 数据集和 ISPDataset 数据集存储在 HDFS 中，使用 Spark Streaming 引擎可以实时读取和写入。

4.3 交叉验证

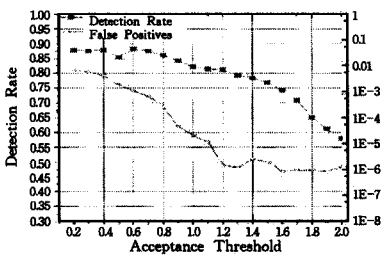
为了评估 BotScanner 系统的检测能力，在标注的基准训练数据集和 LabCapture 数据集上执行交叉验证。两个数据集均来自同一个实验室网络环境，确保了相似的连接环境。对于每个不同的可接受的 α 值，执行了 20 次独立的交叉验证，描述如下：

1)切分基准僵尸家族数据集为训练集合 Tr 和测试集合 Te，分别为 70%和 30%。

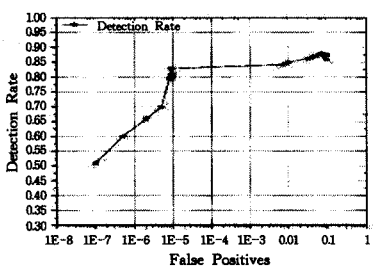
2)假设 LabCapture 数据集是完全干净的，并没有受到恶意代码感染。混合测试集合 Te 与 LabCapture 数据集，生成一个合适的测试数据集用来检测 BotScanner 系统的误报率。

3)然后，使用训练数据集 Tr 训练 BotScanner 检测系统，建立检测模型。

4)最后，使用 Te 与 LabCapture 混合测试数据集检验训练完成的 BotScanner 检测系统。



(a)接受阈值 VS 检测率



(b)误报率 VS 检测率

图 6 BotScanner 系统的检测率和误报率

在实验中，基于已有的僵尸恶意代码样本分析检测结果，如果一个检测特征被正确标识为僵尸恶意代码，则记为准确，否则记为误判。图 6 显示 $\alpha\in[0,2]$的检测率，$h=3$。图 6 中，低的接受率将会产生高于 87%的检测率，但是同时产生了高的误报率。例如，当 $\alpha\leqslant 0.5$ 时，误报率接近 0.1%。正如图 6(a)所示，当检测率接近线性下降时，误报率成指数级下降。当 $\alpha\in[1.2,2.0]$时，与图 6(b)左下角相比，BotScanner 系统取得了较好的检测率和合理的误报率。因此，$\alpha\in[1.2,2.0]$为合适的阈值。对于可接受的阈值 $\alpha=1.2$，系统获得了接近 81.2%的检测率，10^{-6}的误报率。对于此参数，表 2 显示了各个僵尸家族在 5 折交叉验证情况下的检测率。表 2 中，所有的 Conficker 样本检测正确率高达 87%，平均检测正确率高达 81%。误判情况仅仅是 Conficker 和 Black Energy 存在。

表 2 检测率和误报率，BotScanner VS BotHunter ($\alpha=1.2$)

僵尸家族	BotScanner 检测率	BotScanner 误报率	BotHunter 检测率	BotHunter 误报率
Conficker	93%	2	42%	6
Banbra	81%	1	24%	4
Black Energy	74%	3	29%	7
Rustock	83%	1	0	0
Pushdo	79%	2	52%	3
均值	0.82	1.8	0.294	4

由上述可知，BotScanner 系统对于不同的僵尸家族检测率是比较高的。例如，Conficker 家族样本全部显示了高度的周期行为，且非常相似，训练过程中取得了非常高的聚类质量，因此获得了 93%的检测率；Rustock 存在相对低的聚类质量(较高的标准方差)，但是仍然产生了接近相似的行为，获得了 83%的检测率；在恶意代码家族中，Black Energy 获得了最低的检测精度，该家族的聚类簇表现了高度的多样性，聚类簇范围大，聚类质量较低，因此检测精度较低是合理的，而且聚类簇范围大导致了相对高的误报率，出现了 3 次误报。

4.4 模型评估

为了评估 BotScanner 系统，本文在两个数据集上做了大量的实验。第一个数据集是 LabCapture，一个接近 80 台机器的安全实验室 2.5 个月的全包流量数据。根据实验室的管理策略，该环境理论上不存在恶意代码相关的通信流量，可以认为 LabCapture 数据集仅包含良性流量。由于实验室采用的是全包捕获策略，因此可以人工验证已报道的感染和入侵情况。第二个数据集是 ISPDataset，来自运营商，包含 1 个月的网络流量数据。对于规模很大的数据，本文并没有对原始流量进行存储和二次分析，不能够进行精确的检测率比较。本文仅通过将检测到的 IP 地址与已知的受控主机 IP 黑名单列表进行比较，确认受控主机的聚类簇，从而判断本文提出的 BotScanner 系统应用到大型网络的可行性。

在建立 BotScanner 系统原型之后，本文精心准备了以下实验：

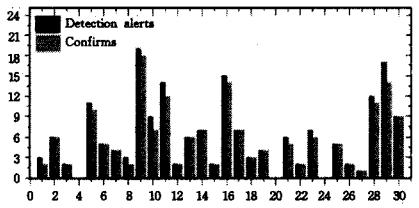
1)基于基准训练数据集和 LabCapture 数据集的交叉验证实验：简言之，训练数据集首先被切分成训练数据集和检测数据集；然后，检测数据集与不包含恶意流量的 LabCapture 数据集混合，记为 HD 数据集(Hybrid Dataset)；当 BotScanner 系统在训练数据集上已经学习到僵尸恶意代码的行为之后，测试程序对准确率和误报率进行分析。由于数据集仅包括僵尸恶意代码的通信流量和 LabCapture 数据集流量，准确率和误报率分析过程比较简单。

2)ISPDataset 数据集上的实验：使用上面提到的 5 类僵尸恶意代码的实际流量训练 BotScanner 系统，并使用 ISPDataset 数据集进行测试，测试过程中 ISPDataset 数据集将以天为单位进行切分测试。BotScanner 系统的告警结果将与僵尸网络受控 IP 黑名单列表进行比较。

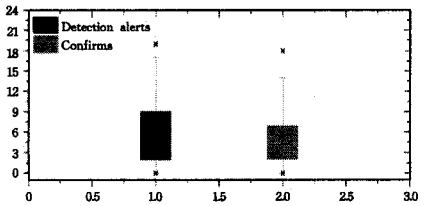
3)与相关工作的比较实验：在本实例中，相关工作中最知名的是基于包检测的 BotHunter 系统，该系统可以从 http://www.bothunter.net/下载。本文在 BotHunter 系统上检测了

实验(1)和实验(2)的测试数据集。

对于实验(1),4.3 节详细描述了实验过程和结果。对于实验(2),实验结果如图 7 和图 8 所示。ISPDataset 数据集为某运营商交换机 1 个月的流量数据,图 7 描述了 BotScanner 检测系统在 30 天的检测结果,并且与黑名单列表进行了比较,平均每天发现 6 个僵尸网络,黑名单列表平均能够确认 4.5 个。图 8 中,对于 BotScanner 的告警信息进行进一步确认,映射到 Conficker、Banbra、Black Energy、Rustock、Pushdo 这 5 个僵尸网络,并与 BotHunter 检测结果进行比较。显然,在实际环境中,BotScanner 总体检测能力优于 BotHunter 系统。值得注意的是,在检测中并没有发现 Black Energy 僵尸网络,这可能存在两个原因,一是在流量中并不存在 Black Energy 的 C&C 通信,另一个是 BotScanner 和 BotHunter 检测系统本身的问题,这将在后续研究中进行确认。

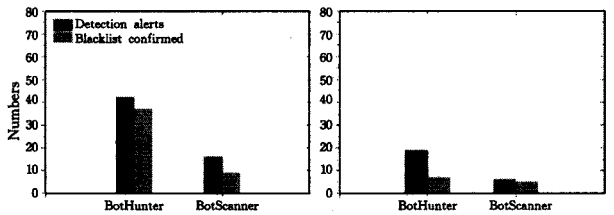


(a)Alerts VS Confirms

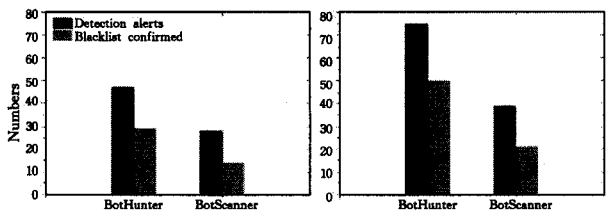


(b)statistics

图 7 ISPDataset 数据集,BotScanner VS Blacklist



(a)Pushdo (b)Rustock



(c)Conficker (d)Banbra

图 8 ISPDataset 数据集,BotScanner VS BotHunter

4.5 检测特征对于检测结果的贡献指数

为了评估 BotScanner 系统的检测算法和检测特征向量各个分量对于检测结果的影响,我们提取了泛化后每个分量对于 γ_M 的贡献,本文称之为贡献指数。各个僵尸家族对于不同特征分量存在着不同的分布:其中,Banbra 和 Conficker 都具有周期性,故能够获得较好的检测质量。对于每个检测分量,其余的僵尸家族在特征分量之间显示了较大的差异性。对于 Pushdo 家族,平均时长和 FFT 的贡献指数较小,间隔和平均流大小具有较大的贡献指数。目标 IP 传输的字节特征对于 Black Energy 并不重要,Rustock 对于该特征完全不敏感。然而,源 IP 字节数,即向 C&C 服务器请求连接,对于 Rustock 检测结果有高度的贡献指数。

对于文献[14]提出的 FFT 分量,该分量在实时流计算中计算复杂度较高,接近 40%。然而,该分量在几类僵尸家族检测中效果明显,具有较高的贡献指数。除了 Conficker 和 Pushdo 这两个僵尸家族,FFT 是其他僵尸家族检测过程中最重要的特征。特别是 Banbra 家族,平均时间特征对于检测贡献指数较小,但是 FFT 特征贡献指数极大。这表明,比起简单的均值特征,潜在频率特征具有更好的周期性和质量。在模型建立过程中也验证了文献[14]的结论,在某些类僵尸网络中使用 FFT 频率特征将具有更高质量的聚类簇。对于所有的 5 个僵尸家族的分析表明,上下行流量比、FFT 特征、平均间隔和 IP 地址熵比其他特征贡献指数相对更大,即平均持续时长和平均流大小贡献指数相对较小。考虑到典型受控主机的行为模式,这个结果更符合其行为过程,即受控主机发送相似的请求给 C&C 服务器,并收到不同大小的回复信息。此外,还观测到潜在的 FFT 频率优于平均间隔均值。

4.6 加速比评估

在本文中,分布式算法效率用加速比(Speed up,Sp)评估。加速比是指数据集固定,对于同一个任务在单节点处理系统和分布式处理系统中运行消耗的时间的比率,用来衡量分布式系统或程序并行化的性能和效果。其中,$Sp = T1/Tp$,Sp 是加速比,$T1$ 是单节点下的运行时间,Tp 是在有 P 个节点的分布式系统中的运行时间。理想情况下,加速比是呈线性的,但实际的加速比要低于理想状态。图 9 是 HD 数据集的加速比,Tp 取 5 次实验的平均值。其中图 9(a)是 BotScanner 检测系统分布式架构选型时测试的 Hadoop、HadoopBinMem 和 Spark 方案,测试算法为本文选用的 Kmeans ‖ 算法,测试过程为第一次迭代。其中,Hadoop 为 1.2 稳定版,HadoopBinMem 是指在首轮迭代中执行预处理,通过将输入数据转换成为开销较低的二进制格式来减少后续迭代过程中文本解析的开销,在 HDFS 中加载到内存。Spark 是指基于 RDD 的系统,在首轮迭代中缓存 Java 对象以减少后续迭代过程中解析、反序列化的开销。显然,在首轮迭代时间上,对于指定的集群规模,Spark 的迭代时间明显优于 Hadoop 和 HadoopBinMem。随着计算节点的增加,这 3 个分布式架构都能够获得一定程度的加速比。对于 Kmeans ‖ 算法,在 10 个计算节点的环境中,3 个分布式架构的首轮迭代时间和后续迭代时间代价如图 9(b)所示。整个检测过程的加速比如图 9(c)和图 9(d)所示。图 9(c)描述了 Netflow 生成过程(SparkG)、主机关系链生成过程及特征提取过程(SparkAC)、主机 Netflow 图谱生成过程及特征提取过程(SparkNG)的加速比,由于 SparkNG 和 SparkAC 过程计算量较大,更多的是 CPU 密集型,因此加速比较 SparkG 相差较多,而且 SparkNG 计算量又高于 SparkAC 过程。对于这 3 个过程,因为节点通信、任务启动、调度等开销,加速比没有达到线性。图 9(d)描述了 Kmeans ‖ 算法获得的加速比,随着节点个数的增多,4 个节点即可完成计算,多个节点的效率并不会提升很多,算法获得的加速比趋于平稳。该实验证实了本文提出的基于 Spark 的僵尸网络分布式实时检测算法更适用于大规模数据集。

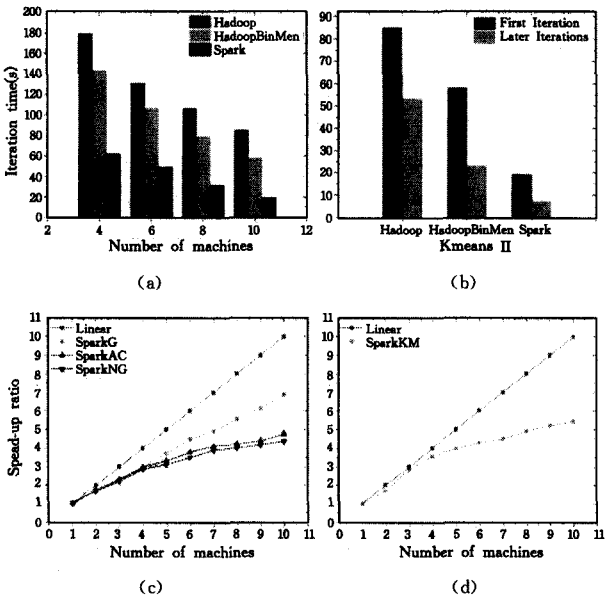


图 9 加速比

结束语 本文展示了一种新颖的僵尸网络分布式实时检测系统 BotScanner，该系统的分布式架构是基于 Spark Streaming 分布式流处理平台，检测特征是基于网络流的统计特征。通过对已知的僵尸家族自动地、非监督地学习之后，BotScanner 系统基于对 C&C 流量的聚类簇统计结果建立检测模型。实验表明，在 C&C 通信具有周期特性的前提下，通过流量模式分析，BotScanner 系统能够获得接近 81% 的检测准确率。文中，最重要的两个创新点如下：其一，在无需 IP 黑名单和深度包解析的前提下，BotScanner 系统能够以较高的准确率检测僵尸网络。因此，BotScanner 系统对于通信加密或混淆的僵尸恶意代码是有效的。其二，基于 Spark Streaming 分布式流处理机制，本文实现了 BotScanner 检测系统的分布式原型。由于引入了分布式流实时处理策略，对于核心交换机上的高速流量，BotScanner 检测系统可以通过添加处理节点的方式轻松处理。因此，BotScanner 系统能够部署到运营商网络中。

BotScanner 系统被看成一个僵尸网络分布式实时检测的框架原型，可在多个层面进行改进，潜在的进一步优化如下：

1)沙箱和虚拟机环境需要进一步改进，以更好地防备僵尸恶意代码作者探测运行环境，停止释放恶意行为。

2)恶意代码家族标注问题，这是反恶意代码研究机构的任务。当前，本文主要是依赖 VirusTotal 和 Anubis 联合标注，即静态检测和动态检测联合判别，能够产生足够好的结果。尤其是在建模的过程中删除了一些特别的、小的聚类簇，从而依赖更多的家族共有的特征。然而，更为精确的分类器能够使得 BotScanner 系统检测能力进一步提高。

3)可以在训练阶段引入非监督的学习方法。据此，选择一种机器学习方法能够获得理想的特征，该特征能够对恶意代码家族进行精确的描述，并在检测步骤赋予合理的权重。也可以尝试使用深度学习方法进行僵尸网络检测。

4)仅使用 Netflow 数据进行检测是本文的一大特点，但 Netflow 数据所含信息量毕竟有限，为进一步提高检测精度、降低误报率，可以尝试使用灵活高效的多源数据采集与融合机制，例如 DNS 数据、信誉数据、进行包解析或关联其他设备的检测结果。

参 考 文 献

[1] Zhuge J W, Han X H, Zhou Y L, et al. Research and development of botnets[J]. Journal of Software, 2008, 19(3): 702-715 (in Chinese)
诸葛建伟，韩心慧，周勇林，等. 僵尸网络研究[J]. 软件学报，2008，19(3)：702-715

[2] Gu G, Porras P, Yegneswaran V, et al. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation[C]// 16th Usenix Security Symposium. 2007

[3] Goebel J, Holz T. Rishi: Identify bot contaminated hosts by irc nickname evaluation[C]// Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets. 2007: 8

[4] Binkley J R, Singh S. An algorithm for anomaly-based botnet detection[C]// Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI). 2006: 43-48

[5] Wurzinger P, Bilge L, Holz T, et al. Automatically generating models for botnet detection[M]// Computer Security-ESORICS 2009. Springer Berlin Heidelberg, 2009: 232-249

[6] Perdisci R, Lee W, Feamster N. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces[C]// NSDI. 2010: 391-404

[7] Giroire F, Chandrashekar J, Taft N, et al. Exploiting temporal persistence to detect covert botnet channels[M]// Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2009: 326-345

[8] Gu G, Zhang J, Lee W. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic[C]// 15th Annual Network and Distributed System Security Symposium (NDSS). 2008

[9] Gu G, Perdisci R, Zhang J, et al. BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection[C]// USENIX Security Symposium. 2008: 139-154

[10] Yen T F, Reiter M K. Traffic aggregation for malware detection [M]// Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Berlin Heidelberg, 2008: 207-227

[11] Strayer W T, Walsh R, Livadas C, et al. Detecting botnets with tight command and control[C]// Proceedings of the 31st IEEE Conference on Local Computer Networks. 2006: 195-202

[12] Nagaraja S, Mittal P, Hong C Y, et al. BotGrep: Finding P2P Bots with Structured Graph Analysis[C]// USENIX Security Symposium. 2010: 95-110

[13] François J, Wang S, Engel T. BotTrack: tracking botnets using NetFlow and PageRank[M]// NETWORKING 2011. Springer Berlin Heidelberg, 2011: 1-14

[14] Tegeler F, Fu X, Vigna G, et al. Botfinder: Finding bots in network traffic without deep packet inspection[C]// Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies. ACM, 2012: 349-360

[15] Coskun B, Dietrich S, Memon N. Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts[C]// Proceedings of the 26th Annual Computer Security Applications Conference. ACM, 2010: 131-140

(下转第 162 页)

World of Computer Science and Information Technology Journal,2012,3(2):105-109

[4] Jamshidi Y, Nezamabadi-pour H. A Lattice based Nearest Neighbor Classifier for Anomaly Intrusion Detection[J]. Journal of Advances in Computer Research,2013,4(4):51-60

[5] Ma Z,Kaban A. K-Nearest-Neighbours with a novel similarity measure for intrusion detection[C]//2013 13th UK Workshop on Computational Intelligence (UKCI). IEEE,2013:266-271

[6] Jianliang M,Haikun S,Ling B. The Application on Intrusion Detection Based on K-means Cluster Algorithm[C]//International Forum on Information Technology and Applications,2009(IFITA '09). IEEE,2009:150-152

[7] Li Z,Li Y,Xu L. Anomaly Intrusion Detection Method Based on K-Means Clustering Algorithm with Particle Swarm Optimization[C]//2011 International Conference on Information Technology, Computer Engineering and Management Sciences (ICM). IEEE,2011:157-161

[8] Deelers S A S. Enhancing K-Means Algorithm with Initial Cluster Centers Derived from Data Partitioning along the Data Axis with the Highest Variance[C]//Proceedings of World Academy of Science,Engineering and Technology. 2007,26:323-328

[9] Gast E,Oerlemans A,Lew M S. Very large scale nearest neighbor search: ideas, strategies and challenges[J]. International Journal of Multimedia Information Retrieval,2013,2(4):229-241

[10] Muda Z, Yassin W, Sulaiman M N, et al. Intrusion detection based on K-Means clustering and Naïve Bayes classification[C]// International Conference on Information Technology in Asia. IEEE,2011:1-6

[11] Ashok R,Lakshmi A J,Rani G D V,et al. Optimized feature selection with k-means clustered triangle SVM for Intrusion Detection[C]//Third International Conference on Advanced Computing. IEEE,2011:23-27

[12] Sharma S K,Pandey P,Tiwari S K,et al. An improved network intrusion detection technique based on k-means clustering via Naïve Bayes classification[C]//2012 International Conference on Advances in Engineering,Science and Management (ICAESM). IEEE,2012:417-422

[13] Muda Z, Yassin W, Sulaiman M N, et al. Intrusion detection based on k-means clustering and OneR classification[C]//2011 7th International Conference on Information Assurance and Security (IAS). IEEE,2011:192-197

[14] Guo C, Zhou Y, Ping Y, et al. A distance sum-based hybrid method for intrusion detection[J]. Applied Intelligence,2014,40(1):178-188

[15] Kuang F,Xu W,Zhang S. A novel hybrid KPCA and SVM with GA model for intrusion detection[J]. Applied Soft Computing,2014,18(4):178-184

[16] Gogoi P, Bhattacharyya D K, et al. MLH-IDS: A Multi-Level Hybrid Intrusion Detection Method [J]. Computer Journal,2014,57(4):602-623

[17] Xiang C,Xiao Y,Qu P,et al. Network Intrusion Detection Based on PSO-SVM[J]. TELKOMNIKA:Indonesian Journal of Electrical Engineering,2013,12(2):1052-1058

[18] Wang Jie-song,Zhang Xiao-fei. The Analysis and Pre-process of KDDCup99 Benchmark Dataset of Network Intrusion Detection [J]. Science and Technology Information,2008(15):79-80(in Chinese)
王洁松,张小飞. KDDCup99 网络入侵检测数据的分析和预处理[J]. 科技信息:科学·教研,2008(15):79-80

[19] Zhang Xin-you, Zeng Hua-shen, Jia Lei. Research of Intrusion Detection system Dataset-KDD CUP99[J]. Computer Engineering and Design. 2010,31(22):4809-4814(in Chinese)
张新有,曾华燊,贾磊. 入侵检测数据集 KDD CUP99 研究[J]. 计算机工程与设计,2010,31(22):4809-4814

[20] Wang Zhi-gang, Hu Chang-zhen, et al. Cyber Security Datasets Research advanced materials research[J]. Advanced Materials and Computer Science II,2013,4(4):191-195

[21] Tsai C,Lin C. A triangle area based nearest neighbors approach to intrusion detection[J]. Pattern Recognition,2010,43:222-229

[22] Elkan C. Results of the KDD'99 classifier learning contest [OL]. http://cseweb.ucsd.edu/users/elkan/clresults.html

(上接第 136 页)

[16] Kheir N,Wolley C. BotSuer:Suing stealthy P2P bots in network traffic through netflow analysis[M]//Cryptology and Network Security. Springer International Publishing,2013:162-178

[17] Fan Y, Xu N. A P2P Botnet Detection Method Used On-line Monitoring and Off-line Detection[J]. International Journal of Security & Its Applications,2014,8(3):87-96

[18] Amini P, Azmi R, Araghizadeh M A. Botnet Detection using NetFlow and Clustering[J]. Advances in Computer Science: an International Journal,2014,3(2):139-149

[19] Garg S,Sarje A K,Peddoju S K. Improved Detection of P2P Botnets through Network Behavior Analysis[M]//Recent Trends in Computer Networks and Distributed Systems Security. Springer Berlin Heidelberg,2014:334-345

[20] Vania J,Meniya A,Jethva H B. A Review on Botnet and Detection Technique[J]. International Journal of Computer Trends and Technology,2013,4(1):23-29

[21] Zhao Y,Xie Y,Yu F,et al. BotGraph:Large Scale Spamming Botnet Detection[C]//NSDI. 2009,9:321-334

[22] Jiang Hong-ling,Shao Xiu-li,Li Yao-fang. Online Botnet Detection Algorithm Using MapReduce[J]. Journal of Electronics & Information Technology,2013,35(7):1732-1738(in Chinese)
蒋鸿玲,邵秀丽,李耀芳. 基于 MapReduce 的僵尸网络在线检测算法[J]. 电子与信息学报,2013,35(7):1732-1738

[23] Batcher K E. Design of a massively parallel processor[J]. IEEE Transactions on Computers,1980,100(9):836-840

[24] Gropp W,Lusk E,Doss N,et al. A high-performance,portable implementation of the MPI message passing interface standard [J]. Parallel Computing,1996,22(6):789-828

[25] Geist A,Beguelin A,Dongarra J,et al. PVM:Parallel virtual machine—a users' guide and tutorial for networked parallel computing[M]. MIT press,1994

[26] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets[C]//Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. 2010:10

[27] Dean J,Ghemawat S. MapReduce:simplified data processing on large clusters[J]. Communications of the ACM,2008,51(1):107-113

[28] Bahmani B,Moseley B,Vattani A,et al. Scalable k-means++ [J]. Proceedings of the VLDB Endowment,2012,5(7):622-633