基于决策树的被动操作系统识别技术研究

易运晖 刘海峰 朱振显
(西安电子科技大学通信工程学院 西安 710071)

摘 要 操作系统识别是网络安全评估的关键技术之一,在网络安全威胁和风险日益加剧的形势下,其研究具有非常重要的意义。针对当前基于TCP/IP协议栈指纹库的操作系统识别技术难以辨识未知指纹所对应的操作系统的问题,提出了基于C4.5决策树模型的被动操作系统识别方法,并将它与其他分类算法进行了比较。通过实验测试验证了分类方法的有效性,并对其结果进行了分析。

关键词 操作系统识别,TCP/IP协议栈,指纹,决策树,有效性
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.8.016

Research of Passive OS Recognition Based on Decision Tree

YI Yun-hui LIU Hai-feng ZHU Zhen-xian
(School of Telecommunication Engineering, Xidian University, Xi'an 710071, China)

Abstract As the problem of network treat is getting worse, it makes great sense to study the method of operation system recognition, which is a key part of network security evaluation. Current operation system recognition based on TCP/IP stack fingerprint database can not recognize unknown fingerprints. A passive operating system identification method based on decision tree was proposed, and it was compared with other classification algorithms. Experiment shows that this classification algorithm owns a better effectiveness and gives the explanation about the result.

Keywords OS recognition, TCP/IP stack, Fingerprinting, Decision tree, Effectiveness

1 引言

随着网络信息的爆炸式增长,信息的收集和分析至关重要,识别远程操作系统的类型具有非常重要的意义。一方面,电脑生厂商可以有效地统计分析他们在市场中所占的份额[1];另一方面,在网络信息安全系统中,识别远程操作系统类型则是其中不可或缺的一个重要组成部分。保障网络安全最大的挑战之一就是及时发现漏洞,而绝大部分安全漏洞和隐患都是与操作系统息息相关的[2]。因此操作系统识别技术正是网络安全评估的关键技术之一,具有很高的研究和应用价值,近年来操作系统识别技术成为了人们研究的热点。

1998年Gordon Lyon[3]发布了基于TCP/IP协议栈的操作系统识别工具Nmap,该工具通过向目标主机发送大量的基于不同协议的数据包,利用目标主机的回应信息去识别远程的操作系统,其最大的不足就是非常容易被入侵检测设备发现,同时指纹库需要手动升级。Greenwald等[4]提出了减少发送网络探针数目的技术,其有效地降低了被入侵检测设备发现的概率。Arkin等[5]提出基于ICMP协议栈指纹特征的主动探测远程操作系统方法,即通过主动发送UDP和ICMP请求报文到目标主机来触发ICMP消息,然后根据ICMP消息中的网络特征值进行基于签名数据库的模糊匹配以及合理的推测,通过模糊矩阵统计分析来探测操作系统,从而确定远程操作系统的类型。该方法使用ICMP取代TCP协议,有效地解决了不同的操作系统使用相同的协议栈带来的识别问题。Medeiros等[6]提出利用TCP协议头部的初始序号来识别操作系统,利用了不同操作系统对建立连接时的初始序号生成和增长模式的不同,该方法需要观测大量的数据包(至少100k),实用性受限,错误率较高。Shamsi等[7]提出基于统计理论的单包操作系统识别模型,在指纹库中加入了RTO(自动请求重传)选项,重传机制使数据包从最初未收到确认响应的那个错误包开始重发,为了能分辨可能收到的重复包,TCP使用唯一的ISN(初始序号)和ACK值来确定应该接收的包,不同操作系统自动请求重传的时间间隔以及重传次数不同。该模型充分考虑了网络的排队时延、丢包率、人为修改协议栈的因素,而且单包技术能有效地降低扫描所需要的流量,减小被入侵检测设备识别的概率。刘英等[8]提出基于TCP/IP协议栈中的TCP可选项的操作系统识别方法,分析了不同操作系统对TCP可选项响应顺序和响应数据的差异,但是该方法不能有效识别未知指纹,正确率有待提高。

随着数据挖掘技术的引入,多种机器学习的方法被应用到操作系统识别技术中。Robert[9]利用了朴素贝叶斯分类器来被动识别操作系统,他利用了两次独立的测试,第一次利用了p0f的指纹库的指纹进行训练,第二次利用自己在Web服务器上收集的指纹进行训练,两次的结果非常类似,有效地处

到稿日期:2015-07-13 返修日期:2015-10-26 本文受国家自然科学基金资助项目(61072067),高等学校科学创新引智计划(2009K01-46)资助。
易运晖(1975-),男,博士,副教授,主要研究方向为网络测量、网络行为学等;刘海峰(1990-),男,硕士生,主要研究方向为网络测量、信息安全,E-mail:liuhaifeng925@126.com(通信作者);朱振显(1990-),男,硕士生,主要研究方向为网络测量。

万方数据

理了未知指纹无法识别的问题，但是此类方法过分依赖于样本在样本空间的分布，具有潜在的不稳性。Carlos 等[10]利用神经网络的方法识别操作系统，该方法基于 Nmap 的数据库，在某一类操作系统识别上，准确率比 Nmap 略高，但是该方法依赖于 RPC 服务，使用范围有限。国内的周铁铮等[11]利用 SVM 实现了操作系统识别，有效地识别了未知指纹对应的操作系统，但并未对分类器输入向量进行降维处理，同时只能粗粒度识别操作系统，识别结果需要进一步细化。四川大学程书宝等[12]提出了基于奇异值分解和使用有向无环图分类的操作系统类型识别新方法，该方法首先对操作系统原始指纹生成的矩阵进行奇异值分解来提取奇异值特征，然后使用有向无环图生成多类分类器对未知指纹的奇异值特征进行分类，该方法有效地降低了向量维数，而且对未知指纹具有较高的识别率。

针对现有技术存在的问题，在现有研究的基础上提出了基于决策树的被动操作系统模型，并将其与其他的分类算法做了比较，最后通过实验验证了模型的有效性。

2 TCP/IP 协议栈指纹分析

互联网协议的标准大多是在 RFC 规范中描述的，但由于 RFC 规范中存在一些可供厂商自由选择、优化改进的可选项，因此不同的操作系统在实现 TCP/IP 协议栈时会存在细微的差别。所谓的 TCP/IP 协议栈指纹识别技术[13]，就是通过探测并分析 TCP/IP 协议栈在实现上的差异来准确辨识远程操作系统的类型。目前，基于 TCP/IP 协议栈指纹匹配是操作系统识别的主流技术。远程操作系统识别主要分为主动和被动[14]两种方式，本文使用被动的操作系统识别方法，与主动的识别方法相比，被动方式不会被入侵检测设备发现，同时不会产生额外的流量开销，而且可以识别防火墙后的主机操作系统，缺点是只能基于已知的网络流量。

为了阻止远程操作系统的探测，网络管理人员可能会阻止远程操作系统的识别，即人为修改注册表的默认参数，同时也出现了网络蜜罐等技术，这些技术增加了远程操作系统识别的难度。这些技术尽管可以有效地修改 IP 数据包的某些默认参数，但是如果缺少技术支持，会产生各种副作用（例如，禁用 TCP 可选项里面的 SACK 可以显著地降低网络的吞吐量）。一些特定的设备（例如，打印机）是不允许 TCP 协议可选项被修改的，而且新的操作系统更倾向于支持较小的可选项的选择，例如，Windows7 提供注册表键禁用时间戳选项，但是修改是不起作用的。这些修改在统计意义上是不重要的。表 1 列出 3 种典型操作系统指纹库里的指纹。

表 1 操作系统指纹

OS	LEN	WIN	DF	TTL	MSS	OPT
Linux2.6	48	5792	1	64	1460	MNWST
Windows7	52	8192	1	64	1460	MNWST
Windows XP	64	65535	1	64	1460	MNWSNT

下面对其中的一些选项进行说明：

(1)LEN 表示 TCP 建立连接时 IP 数据包的长度。

(2)WIN 表示 TCP 建立连接时的窗口大小。

(3)DF 表示是否分片。

(4)TTL 表示生存时间，在接收端无法得到准确的 TTL 初始值，不同的操作系统采用默认 TTL 初始值，典型的有 64,128,255。文献[6]统计出了接收到的数据包的 TTL 值和路由跳数的分布，发现 TTL 值都是分布在 64,128,255 附近，且跳数集中在 15 到 20 跳之间，鉴于此，利用四舍五入的方法推算出操作系统的初始 TTL 值，将指纹库里的 TTL 选项做归一化处理，该处理方式有助于提高操作系统的识别率。

(5)MSS 表示最大报文传输段，此选项表示 TCP 传往另一端的最大数据块的长度，当一个连接建立时，连接的双方都要通告各自的 MSS 值。

(6)OPT 表示 TCP 协议头的可选项，包括窗口扩大选项(WS)、选择确认选项(S)、时间戳选项(T)和无操作选项(N)。可选项是指纹中一个很重要的特征，不同的操作系统可选项的值不同，顺序也不同。可选项部分为通信双方更好地协商通信细节提供了帮助，增强了 TCP 协议通信的可靠性，是识别操作系统的重要依据。

上述指纹都是从 TCP 建立连接时的数据包（SYN 与 SYN+ACK 的数据包）中提取出来的，建立连接时数据包里关于操作系统的指纹最丰富。

3 基于决策树的操作系统识别模型设计

数据分类是数据挖掘的主要内容之一，主要是通过分析数据样本而产生关于类别的精确描述，同时分类技术解决问题的关键在于构造分类器模型。目前分类方法有很多种，主要包括：决策树分类、贝叶斯分类、支持向量机分类方法等。与其他分类方法相比，决策树分类方法有算法简单、计算量小、准确度高以及对训练样本依赖程度低等特点。

3.1 C4.5 决策树模型

3.1.1 信息增益率

信息增益率是用来衡量给定的属性区分训练样本能力的一个重要指标。信息增益率最高的属性将被作为测试属性[15]。决策树的生成过程就是使划分后的不确定性逐渐减小的过程。计算属性的信息增益率的步骤如下所示：假设训练样本集合为 S，并且训练样本被分为 k 类，即为 $C=\{C_1,\cdots,C_k\}$，此时集合 S 的信息熵则为：

$$I(S)=-\sum_{i=1}^{k}p(S_i)\log_2 p(S_i) \tag{1}$$

其中，$p(S_i)$是样本中属于 C_i 的比例。

假设属性集为 A，且 $A=\{A_1,\cdots,A_m\}$，选择 A_j 作为测试属性进行划分，并设 $Values(A_j)$ 为 A_j 的值域，那么属性 A_j 对样本集合 S 的信息增益为：

$$Gain(S,A_j)=I(S)-\sum_{v\in Values(A_j)}\frac{|S_v|}{|S|}I(S_V) \tag{2}$$

其中，$|S|$为样本集合的元素个数，$|S_v|$为 S 中属性 A_j 的值为 v 的子集元素个数。那么就可以计算出分裂信息，其中分裂信息的定义为：

$$SplitInfo(S,A_j)=-\sum_{v\in Values(A_j)}\frac{|S_v|}{|S|}\log_2(\frac{|S_v|}{|S|}) \tag{3}$$

此时便可求出属性 A_j 的信息增益率：

$$Ratio(S,A_j)=\frac{Gain(S,A_j)}{SplitInfo(S,A_j)} \tag{4}$$

训练过程按照上述分支属性选择方法，自顶向下形成决策树分类器。内部节点表示分支属性，叶节点代表类。决策树分类器形成后，从根到叶节点提取合取范式[16]，形成分类规则。

万方数据

3.1.2 决策树剪枝

为了避免决策树自身高度的无限制增长和数据的过度拟合，需要对决策树进行一定的剪枝处理，剪去决策树中可能降低预测准确率的分支。本文采用了后修剪枝的方法，即允许树过度拟合数据，然后在树生成后剪掉子树。决策树剪枝的具体过程如下：对决策树中从根节点开始的所有非叶节点，计算其被剪后的误分类率，将置信区间的上限作为对误分类率的估计。对于一个既定显著性水平度 α（C4.5 算法中将 α 默认为 0.25），错误总数服从二项分布，可得

$$\Pr\left(\frac{|f-q|}{\sqrt{\frac{q(1-q)}{N}}}\geqslant U_{1-\alpha}\right)=\alpha \tag{5}$$

其中，N 是样本实例的总数量，设 E 为 N 个样本实例中被错误分类的个数，那么 $f=\frac{E}{N}$ 为观察到的误差率，q 为真实的误差率。令 $z=1-U_{1-\alpha}$ 为置信度 α 的标准差，通过上式可计算出 q 的置信度上限，然后用此置信度上限为该节点误差率 e 做一个估计：

$$e=\frac{f+\frac{z^2}{2N}+z\sqrt{\frac{f}{N}-\frac{f^2}{N}+\frac{z^2}{4N^2}}}{1+\frac{z^2}{N}} \tag{6}$$

通过判断剪枝前后 e 值的大小，决定是否需要剪枝。

3.2 基于决策树分类器的操作系统识别模型

图 1 所示为本文进行操作系识别的决策树模型。

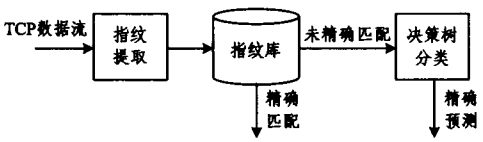

图 1 基于决策树的操作系统识别流程图

首先解析已知的数据包，提取操作系统识别所需要的指纹，并将其与指纹库里面的指纹比较，如果完全匹配，则输出相应的操作系统；若不能完全匹配，则利用决策树分类器进行准确预测。随着越来越多新的操作系统的出现，操作系统指纹也变得越来越丰富，利用机器学习算法对指纹进行分类，可以有效地识别出未知指纹对应的操作系统。在利用决策树进行分类前首先对数据进行预处理，即对采集好的原始数据集进行特征提取、空缺值处理和属性归一化或离散化，并在此基础上形成所需训练数据集。对提取的指纹进行的处理包括初始化 TTL 值，使其取值在集合{64，128，255}里。考虑到 TCP 可选项的顺序差异对识别操作系统的关键作用，不能简单地把可选项当作一个字符串，而需要对可选项部分进行离散处理，把每一项当作一个独立的属性，这样可以体现出可选项的重要价值。预处理完成后，利用训练集建立一棵决策树，构建出决策树模型，利用生成的决策树模型对未知的数据样本进行分类，并进行准确率的预测，同时需要利用一定的测试集对分类模型的准确率等指标进行评估。

4 实验仿真

4.1 实验环境

本文构建的操作系统指纹库部分来自 p0f，Nmap 等主流操作系统数据库的指纹，同时提取了 2015 年 6 月 17 日下午 10:00－12:00 之间经过西安电子科技大学某网络节点网络流量的指纹，一共 2500 个样本，操作系统种类为 10 种。本文的仿真环境是 Weka 平台，其是由新西兰怀卡托大学学者基于 JAVA 开发的开源平台，本身集成了常见的分类模型，如决策树、贝叶斯和支持向量机等，用以实现分类、关联规则、聚类分析和特征过滤等功能。

4.2 实验验证与结果分析

本文主要从整体的识别准确率和建模速度、操作系统识别真正率（TP Rate）和假正率（FP Rate）4 个方面来评估决策树算法的性能。分类的准确率是在数据挖掘分类方法中追求的首要目标，较高的分类准确率将会有更可靠的数据信息供人们参考。建模速度越快，说明算法的计算时间越少，那么效率也就越高；真正率指某类样本正确判断所占的比例值，能够反映出该模型对此类应用的识别好坏。假正率指分类模型把其它应用类型错误地判为实际的操作系统，在涉及风险决策最小化时具有重要作用。同时为了比较决策树模型相对于其他分类器的分类效果，也对支持向量机（SVM）和朴素贝叶斯（(Naïve Bayes)模型进行了仿真。

为了保证算法测试的准确性，采用了十折交叉验证，即将数据集分成 10 份，轮流将其中的 9 份作为训练数据，1 份作测试数据，而这份测试数据即为“未知指纹”，每次验证都会算出相应的正确率，用 10 次结果正确率的平均值作为算法分类的最终正确率。本实验中使用的 10 类操作系统各自的指纹样本数量以及指纹样本的 11 个属性分别如表 2 和表 3 所列。

表 2 操作系统指纹样例数

操作系统	样本数
Windows XP	241
Windows Vista	249
Windows 7	503
Windows 8	117
Windows sever 2003	150
Windows sever 2008	60
Linux 2.2	261
Linux 2.4	179
Linux 2.6	118
Linux 3.0	122

表 3 操作系统指纹样例属性及定义

决策树属性名称	定义
LEN	TCP 建立连接时的 IP 数据包的长度
WIN	TCP 建立连接时的窗口大小
DF	是否分片
TTL	生存时间初始值
MSS	最大报文传输段长度
WS	窗口扩大因子移位数
SACK	选择性确认，使 TCP 只重新发送丢失的包
T	是否含有时间戳选项值
N	是否含有无操作选项，主要用于补充 TCP 头部，使其长度保持为 4 字节的倍数
F	表示该条指纹是从 SYN 包还是 SYN+ACK 数据包中提取的
Optmode	表示可选项的模式，不同操作系统可选项的顺序，可选项支持的个数不同

表 4 列出了操作系统 Linux2.6，Windows XP/Windows 2003 的可选项模式（Optmode），例如 Linux2.6 支持 5 个可选项 MSTNW（M，S，T，N，W 分别对应指纹样例属性里面的 MSS，SACK，T，N，WS 选项），若可选项中没有 W 项，便产生了 MST 顺序以及补位符填充的可选项。不同的操作系统，其可选项的取值、可选项的模式是不同的。

万方数据

表 4　操作系统可选项模式

可选项模式	Linux 2.6	Windows XP/2003
模式 1(都存在)	MSTNW	MNWNNTNNS
模式 2(去掉 S)	MNNTNW	MNWNNT
模式 3(去掉 T)	MNNSNW	MNWNNS
模式 4(去掉 W)	MST	MNNTNNS
模式 5(去掉 ST)	MNW	MNW
模式 6(去掉 SW)	MNNT	MNNT
模式 7(去掉 WT)	MNNS	MNNS

实验结果如表 5 所列，其中朴素贝叶斯(NB)方法整体分类的准确率只有 70.24%，由于其直接使用高斯分布假设而无法有效拟合数据库指纹属性分布，因而分类结果明显较差，准确率相对于 C4.5 和 SVM 较低。朴素贝叶斯模型在处理待分类网络流样本时，首先需要计算出样本属于每种类型的概率，再从中选择概率最大的应用类型作为网络样本的所属类型，计算过程相对复杂。在实际应用场景中，指纹分类问题通常无须频繁重建分类模型，只要求分类模型能够在短时间内处理大量的数据流。而 C4.5 算法在非常短的时间内建立的模型识别准确率是最高的，C4.5 决策树方法不依赖于网络流样本的先验概率，能够有效地避免样本分布变化所带来的消极影响，在处理待测操作系统指纹样本时，仅需进行属性值比较，处理相对简单，在处理大规模指纹样本时具有明显的性能优势。SVM 算法的准确率次之，但是 SVM 建模时间是决策树算法的 10 倍多，处理数据的效率过低，本文需要识别 10 类操作系统，SVM 需要建立 45 个分类器。显然决策树算法最适合解决此类数据的分类问题。

表 5　操作系统分类算法准确率及建模时间比较

算法名称	正确率(%)	错误率(%)	建模时间(s)
C4.5	89.29	10.70	0.06
SVM	81.31	18.69	0.55
NB	70.24	29.76	0.03

表 6 同时包含了 3 种分类算法的平均真正率和平均假正率两个指标，这两个指标能够反映分类模型的好坏。通常追求高的真正率值和低假正率值，特别在有决策风险的环境中，因为错误的肯定所带来的风险比错误的否定更重要。从表中数据看出，C4.5 决策树模型明显优于其他两种分类算法。

表 6　3 种分类算法的识别率

算法名称	平均真正率	平均假正率
C4.5	0.892	0.015
SVM	0.813	0.08
NB	0.702	0.082

表 7 列出了 3 种操作系统识别的结果。对应于 Windows 7 操作系统的 45 条指纹未能被准确识别，但是这些指纹大部分被识别为 Windows Vista，这主要是因为这两种操作系统基于相似的协议栈；Windows sever 2008 的识别率效果比较差，样本数量比较少，在建立模型时缺乏样本有效的分布信息，样本不平衡，是该操作系统识别中面临的重要问题。

表 7　3 种操作系统的识别结果

操作系统	指纹数量	未准确识别指纹数目
Windows 7	503	45
Windows Vista	249	36
Windows sever 2008	60	20

表 5 所列的结果仅为使用多重交叉验证模拟“未知指纹”进行分类的结果，结果验证了决策树算法在分类此类数据时其综合性能有明显的优势。为了测试图 1 所提出的识别系统的效果，有必要使用现实意义上的“未知指纹”输入识别系统进行识别实验。为此，收集了西安电子科技大学某些网络节点的指纹以及 Nmap4.90RC1 数据库的部分“指纹”数据共 621 条，这 621 条指纹相对于所提识别系统是“未知指纹”，将这些“未知指纹”作为图 1 所示识别系统的输入，结果如表 8 所列。

表 8　“未知指纹”的识别结果

测试样本数	指纹库 完全匹配数	决策树 准确预测数	总识别数
621	223(36%)	326(82%)	549(88%)

从表 8 可以看出，“未知指纹“经过指纹库完全匹配的指纹数为 223 条，识别的百分比为 36%，剩余的 398 条指纹经过决策树进行分类，准确预测数为 326，准确率为 82%，图 1 所示的识别系统整体识别准确率为 88%，说明本文提出的识别系统具有较高的识别准确率。

结束语　利用机器学习方法处理操作系统指纹分类问题是近年来操作系统识别一个新兴的研究热点，本文实现了基于决策树的操作系统识别模型，实验证明了该模型的有效性。为了进一步提高识别的准确率，辨识新的操作系统，需要进一步完善指纹库。值得指出的是，由于本文使用了被动操作系统识别技术，只能依据已知的网络数据流，识别的范围有一定的局限性。如何实现主动与被动相结合，以及基于不同协议栈的识别技术，是本文下一步研究的重点。

参 考 文 献

[1] Schwartzenberg. Using Machine Learning Techniques for Advanced Passive Operating System Fingerprinting[D]. Enschchede: University of Twente, 2010

[2] Jiao Jian. A method of identify OS based on TCP/IP fingerprint [J]. International Journal of Computer Science and Network Security, 2006, 6(7B): 77-82

[3] Fyodor. Remote OS Detection Via TCP/IP Stack Fingerprinting [EB/OL]. (2014-06-23)[2014-08-21]. http://insecure.org/nmap/nmap-fingerprinting-article.txt

[4] Greenwald L G, Thomas T. Toward undetected operation system fingerprinting[J]. Proceedings of the first conference on First USENIX Workshop on Offensive Technologies, 2007, 20(8): 6-7

[5] Arkin O. A remote active OS fingerprinting tool using ICMP [J]. USENIX&SAGE, 2002, 27(2): 14-19

[6] Medeiros J, Brito A, Pires P. An Effective TCP/IP Fingerprinting Technique Based on Strange Attractors Classification[C]// Proc. DPM/SETOP. 2009: 208-221

[7] Shamsi Z, Nandwani A, Leonard D. Hershel: Single-Packet OS Fingerprinting[C]// The ACM SIGMETRICS Conference 2014. Austin Texas, ACM Press, 2014: 1-12

[8] Liu Y, Xue Z, Wang Y J. Remete OS Indentification Based on TCP Options[J]. China Information Security, 2007(11): 71-72 (in Chinese)
刘英，薛质，王轶骏. 基于 TCP 协议可选项的远程操作系统识别[J]. 信息安全与通信保密，2007(11)：71-72

[9] Beverly R. A Robust Classifier for Passive TCP/IP Fingerprinting [C]// Proceedings of the 5th Passive and Active Measurement

万方数据

(PAM) Workshop. Boston USA, Springer, 2004: 158-167

[10] Sarraute C, Burroni J. Using Neural Networks to Improve Classical Operation System Fingerprinting Techniques[J]. Electronic Journal of SADIO, 2008, 8(1): 35-47

[11] Zhou Tie-zheng, Li Yuan, Zhang Bo-feng, et al. Operation system recognition based on support vector machines[J]. Journal of Tsinghua University(Science and Technology), 2009, 49(S2): 2164-2168(in Chinese)
邹铁铮,李渊,张博峰,等. 基于支持向量机的操作系统识别方法[J]. 清华大学学报(自然科学版),2009,49(S2):2164-2168

[12] Cheng Shu-bao, Hu Yong. Operating System Recognition based on Singular Value Decomposition and DAG_SVMS[J]. China Information Security, 2013(9): 66-67 (in Chinese)
程书宝,胡勇. 基于奇异值分解和 DAG_SVMS 的操作系统类型识别[J]. 信息安全与通信保密,2013(9):66-67

[13] Shu G, Lee D. A formal methodology for network protocol fingerprinting[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(11): 1813-1825

[14] Kohno T, Broido A, Claffy K C. Remote physical device fingerprinting[J]. IEEE Transactions on Dependable and Secure Computing, 2005, 2(2): 93-108

[15] Tom M. Mitchell. Machine Learning[M]. 增华军,张银奎,译. 北京:机械工业出版社,2013

[16] Liu San-ming, Sun Zhi-xin, Liu Yu-xia. Research of P2P Traffic Identification Based on Decision Tree Ensemble[J]. Computer Science, 2011, 38(12): 26-29(in Chinese)
刘三民,孙知信,刘余霞. 基于决策树集成的 P2P 流量识别研究[J]. 计算机科学,2011,38(12):26-29

(上接第 78 页)

[4] Arumaithurai M, Chen J, Maiti E, et al. Prototype of an ICN based approach for flexible service chaining in SDN[C]// 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2015: 5-6

[5] Gao S, Zeng Y, Luo H, et al. Scalable control plane for intra-domain communication in software defined information centric networking[J]. Future Generation Computer Systems, 2016, 56: 110-120

[6] Liu J, Wang L, Zhang Y, et al. Hierarchical Caching Management for Software Defined Content Network Based on Node Value[M]// Advances in Parallel and Distributed Computing and Ubiquitous Services. Springer Singapore, 2016: 67-73

[7] Cho K, Lee M, Park K, et al. Pack, Wave: Popularity-based and collaborative in-network caching for content-oriented networks [C]// IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2012: 316-321

[8] Arianfar S, Nikander P, Ott J. On content-centric router design and implications[C]// Proceedings of the Re-Architecting the Internet Workshop. ACM, 2010: 5

[9] Psaras I, Chai W K, Pavlou G. Probabilistic in-network caching for information-centric networks[C]// Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking. ACM, 2012: 55-60

[10] Psaras I, Chai W K, Pavlou G. In-network cache management and resource allocation for information-centric networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(11): 2920-2931

[11] Chai W K, He D, Psaras I, et al. Cache "less for more" in information-centric networks (extended version) [J]. Computer Communications, 2013, 36(7): 758-770

[12] Wang Y, Li Z, Tyson G, et al. Design and Evaluation of the Optimal Cache Allocation for Content-Centric Networking[J]. IEEE Transactions on Computers, 2016, 65(1): 95-107

[13] Ren J, Qi W, Westphal C, et al. MAGIC: A distributed MAx-Gain In-network Caching strategy in information-centric networks[C]// 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2014: 470-475

[14] Cai Jun, Yu Shun-zheng, Liu Wai-xi. Caching strategy based on node's importance to community in information-centric networks[J]. Journal on Communications, 2015, 36(6): 173-182(in Chinese)
蔡君,余顺争,刘外喜. 基于节点社团重要度的 ICN 缓存策略[J]. 通信学报,2015,36(6):173-182

[15] Saino L, Psaras I, Pavlou G. Hash-routing schemes for information centric networking[C]// Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking. ACM, 2013: 27-32

[16] Hu X, Gong J. Opportunistic On-path Caching for Named Data Networking[J]. IEICE Transactions on Communications, 2014, 97(11): 2360-2367

[17] Shimizu H, Asaeda H, Jibiki M, et al. Content hunting for in-network cache: Design and performance analysis[C]// 2014 IEEE International Conference on Communications (ICC). IEEE, 2014: 3172-3177

[18] Lee M, Song J, Cho K, et al. Content discovery for information-centric networking[J]. Computer Networks, 2015, 83: 1-14

[19] Zeng Yu-jing, Jin Ming-shuang, Luo Hong-bin. LICA: A Segment-Popularity Based Caching Scheme in ICN[J]. Acta Electronic Sinica, 2016, 44(2): 358-364(in Chinese)
曾宇晶,靳明双,罗洪斌. 基于内容分块流行度分级的信息中心网络缓存策略[J]. 电子学报,2016,44(2):358-364

[20] Wang J M, Zhang J, Bensaou B. Intra-AS cooperative caching for content-centric networks[C]// Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking. ACM, 2013: 61-66

[21] Ge Guo-dong, Guo Yun-fei, Liu Cai-xia, et al. Collaborative Caching and Routing Scheme Based on Local Request Similarity in Named Data Networking[J]. Journal of Electronics & Information Technology, 2015, 37(2): 435-442(in Chinese)
葛国栋,郭云飞,刘彩霞,等. 命名数据网络中基于局部请求相似性的协作缓存路由机制[J]. 电子与信息学报,2015,37(2):435-442

[22] Wang L, Bayhan S, Ott J, et al. Pro-Diluvian: Understanding Scoped-Flooding for Content Discovery in Information-Centric Networking[C]// Proceedings of the 2nd International Conference on Information-Centric Networking. ACM, 2015: 9-18

[23] Xie Z, Li X, Wang X. A new community-based evolving network model[J]. Physica A: Statistical Mechanics and its Applications, 2007, 384(2): 725-732

[24] Newman M. Networks: an Introduction[M]. Oxford University Press, Inc, 2010

[25] Bernardini C, Silverston T, Festor O. Mpc: Popularity-based caching strategy for content centric networks[C]// 2013 IEEE International Conference on Communications (ICC). IEEE, 2013: 3619-3623

万方数据