基于KNN和GBDT的Web服务器指纹识别技术

南世慧[1] 魏 伟[2] 吴华清[1] 邹金蓉[2] 赵志文[1,2]
(北京师范大学研究生院珠海分院 广东 珠海 519087)[1]
(北京师范大学信息科学与技术学院 北京 100875)[2]

摘 要 现有的Web服务器指纹识别方法容易因响应头被篡改而得不到准确的识别结果,而且已有的基于机器学习的相关识别方法需要预先发送大量的请求来进行识别。针对上述问题,通过分析响应头的特征关系,提出一种基于KNN和GBDT的Web服务器指纹识别算法,其只需要发送两种不同类型的异常请求,就能识别对应的Web服务器指纹类型和版本范围。与已有Web服务器指纹识别算法进行的对比实验结果表明,所提算法的识别速度和准确率均得到了优化。

关键词 Web指纹,梯度提升决策树,集成学习,网络安全
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.08.025

Web Server Fingerprint Identification Technology Based on KNN and GBDT

NAN Shi-hui[1] WEI Wei[2] WU Hua-qing[1] ZOU Jing-rong[2] ZHAO Zhi-wen[1,2]
(Zhuhai Branch, Graduate School of Beijing Normal University, Zhuhai, Guangdong 519087, China)[1]
(School of Information Science and Technology, Beijing Normal University, Beijing 100875, China)[2]

Abstract Conventional Web server fingerprinting method is easy to modify the response head so that the recognition result is not accurate, and the existing recognition method based on machine learning needs to send a large number of requests for identification. To solve these problems, by analyzing the feature relations of the response head, a Web server fingerprint recognition algorithm based on KNN and GBDT was proposed. Only two different types of exception requests are sent to identify the corresponding Web server fingerprint type and version range. Compared with the existing algorithm of the relevant Web server fingerprint recognition, the proposed algorithm can optimize the recognition speed and the recognition accuracy.

Keywords Web fingerprint, Gradient decision boosting tree, Ensemble learning, Cyber security

1 引言

随着互联网的不断发展,Web应用已成为我们生活中不可或缺的一部分。互联网的触手可及,使得攻击者的攻击手段更加简单,攻击成本不断下降[1],他们在安全的相关社区和论坛就可以检索到很多攻击工具[2-3]来发动攻击。近年来,攻击者根据Web指纹信息,使用对应的Web应用漏洞脚本进行攻击,造成了恶劣影响,且攻击案例逐年增加。因此,准确地识别Web服务器指纹信息,及时地了解Web服务可能存在的安全漏洞,并且对Web服务进行隐患排查和防护[4],具有重要的意义。已有的Web服务器指纹识别工具[5]出于对速度的考虑,大部分依旧使用返回响应头Server等相关字段信息(如Wappalyzer,Wh-atweb)的方式,如果这些字段被篡改,就无法准确识别对应的产品信息。而现有的研究通过机器学习的方法,需要发送大量的异常请求信息,然后统计、比对,进行训练识别,该过程较复杂。

为了使Web服务器指纹识别的准确度和速度得到保证,本文分析了大量不同请求返回的响应头的相关字段信息,经过实验选取了具有明显特征的两种异常请求,并在此基础上融合了KNN和GBDT算法,设计并实现了一个Web服务器指纹识别系统。本文的主要贡献如下:

1)使用两种具有特征性的异常请求,避免使用大量请求。在特征的选取上,与已有方法不同,所提方法添加了响应头顺序长度以及状态码。

2)使用两种机器学习模型嫁接,设计并实现了基于集成学习的Web服务器指纹识别系统。

2 相关工作

网络空间指纹[5]指不同互联网设备开放端口的相关信息,具体包含设备指纹以及Web指纹两类。设备指纹包含开放的端口、端口产品、端口协议、版本等;Web指纹包含应用名、前端框架、后端框架、服务器语言、中间件等信息。常见的端口扫描方法包括主动嗅探以及被动嗅探。主动嗅探指计算机通过自主构造一个有针对性的数据包进行探测;被动嗅探

到稿日期:2017-05-09 返修日期:2017-10-13
南世慧(1993-),男,硕士生,主要研究领域为Web安全;魏 伟(1993-),女,硕士生,主要研究领域为移动安全;赵志文(1966-),男,博士,教授,博士生导师,主要研究领域为信息安全、云计算机安全,E-mail:zhaozw126@126.com(通信作者)。

指计算机仅通过检测周边链路的流量进行判断。

对于 Web 服务器指纹，Lee[6] 最早提出通过发送一个找不到的 URL 来比对返回网页的正文内容，例如 Apache 返回的内容是“not found”，而 IIS 服务器返回的内容会在 Apache 返回的内容前面多加一个 object 来发现不同服务器的差异。2015 年，Wu 等[7] 通过贝叶斯的方法构造了 15 个异常请求，并以此来训练和分析服务器的类型版本。而 Huang 等[8] 认为从返回头的信息中直接获取 Server 信息是最简单、最快捷的方法，但是相关信息容易被篡改，需要通过异常请求的返回信息进行统计分析，并且只能获得服务器类型，不能获得版本范围。2016 年，Yan 等[9] 根据统计的结果，直接使用固定规则来区分服务器的类型。同年，Cao 等[10] 提出使用 K-means 对响应头的头字段数进行分析，以识别网络空间的终端设备。

目前常见的 Web 指纹识别工具有 HMAP 和 Httpprint。HMAP 大量地构造不同长度的 URL，对同一服务器、不同 URL 返回的响应码和响应长度进行统计分析，并将它们组合成指纹库；然后通过匹配判断对应的服务器所属的类型。Httpprint 通过统计的方法对大量服务器的返回信息进行签名并保存，当需要检测服务器时，将现有的签名信息与签名库中的信息进行匹配统计，然后得出响应的结果。

已有研究中，Wu 等[7] 的基于贝叶斯的方法虽然能够识别服务器的类型版本，但是识别的过程中需要构造 15 个异常请求，会花费大量时间。闫淑筠等[9] 所提的固定规则虽然简单，但是在实测过程中存在误报的现象。曹来成等[10] 所提方法可以较好地识别终端设备，但是对服务器的识别不能达到很好的效果，因为对于一个固定的请求，不同服务器返回的头字段的个数和类型很可能一样。

因此，对 Web 服务器的指纹进行研究是有必要的。在此基础上，本文提出基于集成学习的 Web 服务器指纹识别方法，该方法只需要两个请求就能获得 Web 服务器的类型和版本范围。本文方法通过获取网络的 Web 指纹，将 KNN 和 GBDT 算法组成一个模型，并使用该模型对 Web 服务器进行识别。

3 基于集成学习的分类模型

本文使用的算法涉及集成学习算法 GBDT。集成学习[11] 是监督学习的算法，通过组合其他弱分类的机器学习算法形成一个强分类的机器学习算法。常见的形式有使用弱分类的机器学习算法对数据的随机子集进行训练，然后通过算法选择出最合适的分类结果。众所周知，该方法能克服数据过拟合的问题，减少系统的泛化，特别是对高度不平衡的数据集尤为适用。

3.1 梯度提升决策树

梯度提升决策树[12]（GBDT）已经被成功地应用于许多领域，如智能城市，其主要优点是具有通过最小误差和决策树学习自动找到非线性相互影响的能力。GBDT 通常被认为是最好的开箱即用的分类器之一，已在机器学习领域逐渐得到普及，它有能力将弱势学习者整合成一个强大的学习者。

对于多分类问题，GBDT 将损失函数定义为：

$$\Psi(\{y_g,R_g(X)\}_1^G)=-\sum_{g=1}^{G}y_g\log p_g(X) \tag{1}$$

其中，$p_g(x)=P(y_g=1|x_g)$，而且 $p_g(x)$ 和 $R_g(x)$ 的关系为：

$$R_g(x)=\log p_g(x)-\frac{1}{G}\sum_{g=1}^{G}\log p_g(x) \tag{2}$$

则可以得到梯度：

$$\begin{aligned}\tilde{y}_{ig}&=-\left[\frac{\partial\Psi(\{y_{ij},R_j(x_i)\}_{j=1}^G)}{\partial R_g(x_i)}\right]_{\{R_j(x)=R_{j,m-1}(x)\}_1^G}\\&=y_{ig}-p_g(x_i)\end{aligned} \tag{3}$$

根据式(3)，需要计算 G 个参数以及 $R_g(x)$，其中 $R_g(x)$ 可以理解为 x 属于 g 类的概率。

针对 G 个分类，具体的算法流程如下。

算法 1 GBDT 伪代码

$R_{g0}(x)=0,k=1,\cdots,G;$

For m=1 to M do:

$p_g(x)=\exp(R_g(x))/\sum_{g=1}^{G}\exp(R_g(x)),g=1,\cdots,G;$

For g=1 to G do:

$\tilde{y}_{ig}=y_{ig}-p_g(x_i),i=1,\cdots,N;$

$\{T_{glm}\}_{l=1}^{L}=L-$termial node tree$(\{\tilde{y}_{ig},x_i\}_1^N);$

$$\gamma_{glm}=\frac{G-1}{G}\frac{\sum_{x_i\in T_{glm}}\tilde{y}_{ig}}{\sum_{x_i\in T_{glm}}|\tilde{y}_{ig}|(1-|\tilde{y}_{ig}|)},l=1,\cdots,L;$$

$R_{gm}(x)=R_{g,m-1}(x)+\gamma_{glm}1(x\in T_{glm});$

endFor

endFor

通过算法 1，最终得到 $\{R_{gm}(x)\}_1^G$，将其映射成对应的类型概率 $\{p_{gm}(x)\}_1^G$，生成分类的规则。

$$\hat{g}(x)=\arg\min_{1\leqslant g\leqslant G}\sum_{g'=1}^{G}d(g,g')p_{g'm}(x) \tag{4}$$

以式(4)中的 $d(g,g')$ 为代价函数，其表示实际值为 g'、估计值为 g 时的代价。

3.2 最近邻法

最近邻法（KNN 算法）[13] 是一种非参数统计方法，常用于分类和回归问题中。其输入由 K 个最接近的训练样本集合组成，输出是一个分类族群。一个对象最终被确定分到哪一类是由其附近的节点投票确定的。如果 K 值为 1，表明该对象的类别由最近一个节点决定。在回归问题中，KNN 输出这个对象的属性值，并且该值为其 K 个最近邻的平均值。最近邻法根据向量空间模型进行分类，相同类别事物的相似度高，因此可以通过计算未知类别与已知类别的相似度来评估其可能的分类。KNN 是一种局部相似的将所有计算推迟到分类之后的惰性学习算法。其具体步骤如下：

1）确定参数 K，K=最近邻居数；

2）计算并查询实例与所有训练样本之间的距离；

3）对距离进行排序，根据 K 个最小距离确定最近的邻居；

4）确定最近邻居的 Y 个类别；

5）使用大多数最近邻居的类别作为查询实例的预测值。

3.3 KNN 和 GBDT 的混合模型

服务器指纹信息比较抽象，想要获得版本信息，需要先确定服务器的类型。因此，本文在处理数据时，先将数据分为一个大类，然后再根据这个大类确定具体的版本。

为了减小模型错误判断服务器类型的影响，本文使用两种模型对服务器类型和版本范围进行识别。首先用 KNN 模型对服务器类型进行识别，然后将这个模型的结果和其他数据输入 GBDT 模型，最终得出一个版本结果。由于有其他数

据作为干预，即使对服务器版本的判断是错误的，对版本范围的判断也不一定是错的。

本文将 KNN 和 GBDT 相结合，其具体结构如图 1 所示。

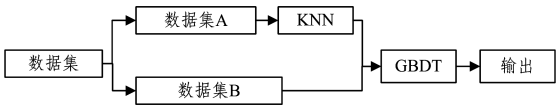

图 1　KNN 和 GBDT 的混合模型

Fig. 1　KNN and GBDT hybrid model

由图 1 可知，先根据原始数据的部分属性，用 KNN 算法获得一个弱分类；然后在这个分类中加入其他数据的部分属性，并对数据施加 GBDT 算法，最终获得一个输出。其中，数据集 A 是异常操作请求“HEAD/HT/1.1\r\n\r\n”的返回信息，而数据集合 B 是异常版本请求“PUT/HTP/ 3.5 \r\n\r\n”的返回信息。

4　基于集成学习的 Web 服务器指纹识别模型的设计与实现

本节将上一节的模型应用到 Web 服务器指纹识别中，以进一步提高识别的准确性，并对其进行实现。

4.1　数据的采集

为了能够使用尽量少的特征来获得更加明确的结果，所选取的 Web 服务器指纹的特征需要具有代表性，一些可以被修改的字段不适合作为特征。因为 RFC 文档没有明确地对异常请求进行规范，所以不同的 Web 服务器对异常请求的处理有细微的差别。因此，本文对异常请求的返回字段进行特征提取和分析。

通过前期大量抓取原始数据并分析，可以发现不同 Web 服务器返回的报文的头信息顺序不一样。对于一个 GET 请求，不同的服务器返回的报文如图 2、图 3 所示。

```
root@kali:~# curl -i 1.202.219.45
HTTP/1.1 200 OK
Server:Apache-Coyote/1.1
Set-Cookie:JSESSIONID=5DC8B436F0A666E81920D31F36C4A1F6;
Path=/
Content-Type:text/html;charset=gb2312
Content-Length:2319
Date:Tue,25 Apr 2017 07:53:08 GMT
```

图 2　Apache 服务器的响应报文

Fig. 2　Response message of Apache server

```
root@kali:~# curl -i 130.185.74.2
HTTP/1.1 200 OK
Content-Type:text/html
Last-Modified:Wed,03 Aug 2016 14:34:18 GMT
Accept-Ranges:bytes
ETag:"d326371e94edd11:0"
Server:Microsoft-IIS/8.5
X-Powered-By:ASP.NET
Date:Tue,25 Apr 2017 07:49:57 GMT
Content-Length:701
```

图 3　Microsoft IIS 服务器的响应报文

Fig. 3　Response message of Microsoft IIS server

通过对比大量的数据可以发现，不同的服务器的 Content-Type，Server，Date，Content-Length，Connection 5 个字段的顺序，与字段是否存在，存在相关性。此外我们还发现，不需要构造太多的请求内容就可以识别不同服务器的类型。经过实验，本文使用如下两种类型的请求便可以达到识别服务器以及版本范围的目的。

“HEAD/HT/1.1\r\n\r\n”

“PUT/HTP/3.5\r\n\r\n”

本文通过以上两种异常请求来收集数据，进而确定服务器类型和版本范围。

4.2　Web 服务器类型的识别

通过向目标服务器发送“HEAD/HT/1.1\r\n \r\n”请求，获得 Content-Type，Server，Date，Content-Length，Connection 5 个字段的位置，此处定义数据的维度为 5，各维度依次表示以上 5 个字段在请求中的相对位置。如果字段不存在，则在对应的字段上置 0。其特征如表 1 所列。

表 1　Web 服务器类型的识别属性

Table 1　Identification properties of Web server type

特征	含义
Content-Type	Content-Type 在响应头的位置
Server	Server 在响应头的位置
Date	Date 在响应头的位置
Content-Length	Content-Length 在响应头的位置
Connection	Connection 在响应头的位置

以上 5 个维度的取值范围为 0～5。为了避免单个属性的权重影响过大，对 5 个属性进行正规化。

4.3　Web 服务器版本范围的识别

不同于对服务器类型的识别，在对服务器的版本进行识别时，发送的请求为“PUT/HTP/3.5 \r\n\r\n ”，除了获取对应服务器的 Content-Type，Server，Date，Content-Length，Connection 字段的顺序位置，还需要状态码以及 Content-Length 的具体长度。另外还需要从 KNN 模型获取识别结果的字段。因此，其维度为 8，具体如表 2 所列。

表 2　Web 服务器版本范围的识别属性

Table 2　Identification properties of Web server version scope

特征	含义
状态码	响应头状态码
Length	Content-Length 的具体数值
KNN_label	KNN 分类器的结果
Content-Type	Content-Type 在响应头的位置
Server	Server 在响应头的位置
Date	Date 在响应头的位置
Content-Length	Content-Length 在响应头的位置
Connection	Connection 在响应头的位置

由于状态码的数值和 Content-Length 的长度过大，因此需要进行一次正规化，使每个维度的数量级相同。

4.4　Web 服务器指纹识别的实现

根据上文的算法模型，使用 Python 语言对 Web 服务器指纹模型进行实现，并在此基础上加入对 Banner 的抓取，通过多种因素来提高模型的准确率。在调度上，由于大部分请

求是 IO 频繁的，因此采用多进程与多协程的组合来加快识别的速度。

整体的模块包含数据采集模块、服务器类型识别模块和版本范围识别模块。融合 Banner 信息后，识别的具体流程如图 4 所示。

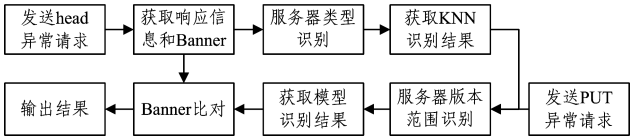

图 4 Web 服务器识别模块的流程图

Fig. 4 Flowchart of Web server identification module

识别的主要步骤如下：

1)发送“HEAD/HT/1.1\r\n\r\n”异常请求，获取结果；

2)将返回的结果进行预处理，输入服务器类型识别模型；

3)发送“PUT/HTP/3.5\r\n\r\n”异常请求，获取结果；

4)对返回的结果以及输入的服务器类型识别模型结果进行预处理，再将处理结果输入到服务器版本识别模型中；

5)获取版本识别结果，将其与 Banner 条进行比对，优先使用模型的结果。

5 实验结果及分析

为了更好地确定识别效果，在实验中使用多种机器算法进行比对。用于测试的标签有 Microsoft IIS，Apache 以及 Nginx。数据总量为 3000 多条，将其中的 400 条作为测试集。

此处对服务器的版本范围做如下设定(见表 3)。

表 3 Web 服务器版本范围的定义

Table 3 Definition of Web server version scope

服务器类型	版本范围
Microsoft IIS	5.0,6.0,7.0－8.0
Apache	1.X,2.X
Nginx	<1.5.4,>1.5.5

5.1 实验数据来源

实验数据来源于 Sodan 和 Zoomeye。Sodan 是国外的网络空间指纹搜索引擎，而 Zoomeye 是国内的网络空间搜索引擎。随机从以上两个搜索引擎中检索出相关数据用于实验，并通过人为预先打标进行预处理，以保证其准确性。

5.2 服务器类型的实验结果

将随机的 2593 条 IP 数据作为训练集，613 条 IP 数据作为测试集。对于相同的特征，使用不同的机器学习算法进行训练，结果如表 4 所列。

表 4 服务器类型的识别结果

Table 4 Identification results of server type

模型	准确率/%	耗时/s
SVM	97.06	0.297
Neural Network	97.06	0.417
KNN	98.03	0.007
SGD	97.88	0.009
Logistic Regression	98.21	0.048
Decision tree	97.88	0.0009

通过实验结果发现，对于服务器类型的模型识别，KNN 算法在综合速度和准确率上相对较好，在理论上比较成熟且简单，适合非线性的数据，并且由于服务器类型这种分类问题是有界的，因此最终采用了 KNN 算法。

5.3 服务器范围的实验结果

将训练好的 KNN 模型作为服务器范围识别的部分数据输入，同样地，将 2593 条 IP 数据作为训练集，将 613 条 IP 数据作为测试集，使用不同模型组合进行测试，结果如表 5 所列。

表 5 服务器版本范围的总体识别结果

Table 5 Overall identification results of server version range

模型	准确率/%	耗时/s
KNN＋SVM	79.66	0.265
KNN＋NeuralNetwork	79.90	0.405
KNN＋KNN	81.36	0.007
KNN＋SGD	72.64	0.011
KNN＋Logistic Regression	79.66	0.068
KNN＋Decision tree	89.83	0.001
KNN＋GBDT	90.31	2.881

通过表 5 的数据可以发现，相比于其他模型，KNN＋GBDT 模型在识别服务器版本范围时的准确率较高，超过 90%，而其他算法的准确率在 80%左右。在耗时上，KNN＋GBDT 明显耗时较长，因为 KNN＋GBDT 在构建树的过程中，每一棵新树都是用于纠正以前训练过的树所产生的误差，因为树的生成是依次构建的，所以需要较长的时间。但是，耗时较长的问题可以通过协程技术得以缓解，因为在具体的识别场景中，需要先发起请求以获得返回报文，然后再进行识别，而等待请求返回将花费大量时间。通过预先将请求添加到协程池，避免了 IO 阻塞。通过实验对 115.47.1.1/24 网段的主机的识别速度进行对比，结果如表 6 所列。可以发现，加入协程技术后，识别的速度有了很大的提高。加入协程技术前，算法是串行识别的；而加入协程后，算法是并行识别的，因此识别速度较快。

表 6 融入协程技术前后 KNN＋GBDT 检测耗时的对比

Table 6 Comparison of detection time before and after using coroutine technique

识别方法	识别主机数量	耗时/s
KNN＋GBDT	255	251
协程＋KNN＋GBDT	255	31

表 7 给出了不同服务器版本的具体准确率。通过分析可得，不同算法的差别主要体现在 Apache 以及 Nginx 服务器的识别上，KNN＋GBDT 算法的识别率相对比较稳定。

表 7 不同服务器版本范围的识别结果

Table 7 Recognition results of different server version range

模型	准确率/%		
	Microsoft IIS	Apache	Nginx
KNN＋SVM	100	89.71	50.00
KNN＋NeuralNetwork	100	90.44	50.00
KNN＋KNN	100	64.71	87.96
KNN＋SGD	100	42.65	68.52
KNN＋Logistic Regression	100	89.71	50.00
KNN＋Decision tree	100	91.18	87.96
KNN＋GBDT	100	91.91	88.92

在识别速度上，对 115.47.1.1/24 网段的主机识别结果进行对比。经手工验证可知，该网段的存活主机有 9 个，且存活主机的主页返回内容都是“400 bad request”或者“502 bad gateway”。

Whatweb 对 Web 指纹的抓取依赖于 Banner 信息，因此能够识别的数量较少；Httpprint 依赖于指纹库，目标主机返回的信息没有“200 OK”的相关信息，因此准确率较低；本文程序通过不同服务器对异常请求返回报文的特征进行识别，即使没有返回消息“200 OK”，返回的文本中也依旧有特征，因此识别率较高。各工具对服务器指纹的识别情况如表 8 所列。

表 8　服务器指纹识别的检测耗时

Table 8　Time consuming of server fingerprint identification

工具	数量	准确识别数量	耗时
Whatweb	255	3/9	6min49s
Httpprint	255	2/9	2min5s
本程序	255	9/9	31s

可以很直观地发现，加入协程技术后，在恶劣复杂的环境下（无法访问、超时），程序的检测速度很快。

相对于多线程，协程技术（即微线程）在协程之间相互切换时，不需要借助操作系统控制，而直接在用户态进行切换，避免了因陷入内核级别的切换而产生性能上的损失。因此，对于 IO 频繁的操作，使用协程技术的识别速度更快。

结束语　本文将集成学习算法 GBDT 与 KNN 算法相结合，设计并实现了一个可以识别 Web 服务器指纹的模型。该模型不需要预先发送大量的请求信息，仅需要发送两个请求就能识别出服务器的类型以及版本范围。通过实验表明，相比于其他典型的识别算法，本文模型具有很好的识别率，同时也避免了以往采用机器学习模型使用较多请求才能识别 Web 服务器指纹的问题，在保证识别速度的同时兼顾了准确率。此外，该系统采用了协程技术，提高了 IO 频繁行为的操作效率，在面对大量数据的检测服务请求时，也能够以较快的速度进行识别。

对于具体的版本号，只利用本模型无法做到准确识别，后续需要进一步分析同一类服务器的不同版本子号在不同异常请求下的区别，以更好地判断详细的版本号。

参 考 文 献

[1] LI F, DURUMERIC Z, CZYZ J, et al. You've got vulnerability: Exploring effective vulnerability notifycations[C]// Proceedings of the 25th USENIX Security Symposium. 2016: 1033-1050.

[2] MAKINO Y, KLYUEV V. Evaluation of Web vulnerability scanners[C]// IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. IEEE, 2015: 399-402.

[3] PARVEZ M, ZAVARSKY P, KHOURY N. Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities[C]// IEEE International Conference for Internet Technology and Secured Transactions. 2015: 186-191.

[4] KISS B, KOSMATOV N, PARIENTE D, et al. Combining Static and Dynamic Analyses for Vulnerability Detection: Illustration on Heart bleed[M]// Hardware and Software: Verification and Testing. 2015.

[5] KHADEMI A F, ZULKERNINE M, WELDEMARIAM K. An Empirical Evaluation of Web-Based Fingerprinting[J]. Software IEEE, 2015, 32(4): 46-52.

[6] LEE D, ROWE J, KO C, et al. Detecting and Defending against Web-Server Fingerprinting[C]// 2002 Proceedings Computer Security Applications Conference. IEEE, 2002: 321-330.

[7] WU S H, SUN D, HU Y. Web Server Identification Based on Bayesian Theory[J]. Computer Engineering, 2015, 41(7): 190-193, 198. (in Chinese).
吴少华，孙丹，胡勇. 基于贝叶斯理论的 Web 服务器识别[J]. 计算机工程，2015，41(7)：190-193，198.

[8] HUANG Z, XIA C, SUN B, et al. Analyzing and summarizing the web server detection technology based on HTTP[C]// 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE, 2015: 1042-1045.

[9] YAN S J, WANG W J, ZHANG Y Q. An efficient method of Web fingerprint identification[J]. Journal of University of Chinese Academy of Sciences, 2016, 33(5): 679-685. (in Chinese)
闫淑筠，王文杰，张玉清. 一种有效的 Web 指纹识别方法[J]. 中国科学院大学学报，2016，33(5)：679-685.

[10] CAO L C, ZHAO J J, CUI X, et al. Cyberspace device identification based on K-means with cosine distance measure[J]. Journal of University of Chinese Academy of Sciences, 2016, 33(4): 562-569. (in Chinese)
曹来成，赵建军，崔翔，等. 基于余弦测度下 K-means 的网络空间终端设备识别[J]. 中国科学院大学学报，2016，33(4)：562-569.

[11] MAHDAVI A. Applying an Ensemble Learning Method for Improving Multi-label Classificati on Performance[C]// 2016 2nd International Conference of Signal Processing and Intelligent System(ICSPIS). 2016.

[12] LIAO Z, HUANG Y, YUE X, et al. In Silico Prediction of Gamma-Aminobutyric Acid Type-A Receptors Using Novel Machine-Learning-Based SVM and GBDT Approaches [J]. Biomed Research International, 2016, 2016(6): 1-12.

[13] LIU Q, LIU C. A novel locally linear KNN model for visual recognition[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2015: 1329-1337.