文章编号:2095-6134(2016)05-0679-07

一种有效的 Web 指纹识别方法*

闫淑筠[1,3†], 王文杰[1,2], 张玉清[1,2,3]

(1 中国科学院大学计算机与控制学院, 北京 101408; 2 中国科学院信息工程研究所信息安全国家重点实验室, 北京 100093;
3 中国科学院大学国家计算机网络入侵防范中心, 北京 101408)

(2016 年 2 月 19 日收稿; 2016 年 4 月 1 日收修改稿)

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.

摘 要 准确获取 Web 服务器及其承载的应用的类型及版本对 Web 站点的安全测试有重要意义. 针对 Web 服务器 Banner 易被修改, 提出使用黑盒测试方法对主流 Web 服务器进行分析,进而选取可有效防止 Banner 欺骗的 Web 服务器指纹; 针对 Web 应用关键字易被删除, 提出使用源码审计方法对主流开源 Web 应用进行分析, 进而选取与其功能相关的 Web 应用指纹, 并构建 Web 指纹库. 在此基础上, 设计并实现 Web 指纹识别工具——WebEye. 实验结果表明, 与主流工具相比, WebEye 能更快速准确地识别 Web 服务器及应用, 并具有良好的可扩展性.

关键词 Web 服务器; Web 应用; Web 指纹识别; Web 指纹库

中图分类号:TP393 **文献标志码**:A **doi**:10.7523/j.issn.2095-6134.2016.05.016

An efficient method of web fingerprint identification

YAN Shujun[1,3], WANG Wenjie[1,2], ZHANG Yuqing[1,2,3]

(1 *School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing* 101408, *China*;
2 *State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093, *China*;
3 *National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing* 101408, *China*)

Abstract It is very important to accurately acquire information of the web server and deployed application for website security testing. Since the web server's Banner was apt to be modified, we used the black-box testing method to analyze major web servers, and then selected web server's fingerprint which could prevent Banner cheating. Since the web application's keywords were apt to be deleted, we used the source code audit method to analyze major web applications, and then selected web application's fingerprint, which was associated with its function, and built a web fingerprint database. Furthermore, a web fingerprint identifying tool WebEye was designed and implemented. Experimental results show that WebEye faster and more accurately identifies the web server and application than similar tools, and it has good scalablity.

Key words web server; web application; web fingerprint identification; web fingerprint database

* 国家自然科学基金(61572460, 61272481)、信息安全国家重点实验室开放课题基金(2015-MS-06)和 360 项目资助
† 通信作者,E-mail:yansj@nipc.org.cn

Web 指纹是 Web 服务组件在开发时留下的可对其类型及版本进行标识的特殊信息，包括 Web 服务器指纹、Web 应用指纹以及前端框架指纹等[1]. Web 指纹识别是指利用已有的 Web 指纹库实现对 Web 服务组件相关信息的准确识别. Web 指纹识别技术可用于 Web 渗透的信息收集环节，准确获取 Web 服务组件的类型及版本信息可有效检测其存在的安全弱点. 大多 Web 攻击主要利用 Web 服务器及应用存在的安全弱点，进一步利用相应的攻击手法获取目标站点的高级权限和敏感数据[2-4]. 因此，本文主要研究对 Web 站点安全有重要意义的 Web 服务器及应用的指纹识别技术.

国内外对 Web 服务器及应用指纹的研究[5-6]通过构造大量特殊的 HTTP 请求与 Web 服务器交互，从其响应报文中提取 Web 服务器及应用的指纹. 然而，由于研究工作涉及较复杂的 Web 指纹识别流程，在实际应用中，主流工具如 WhatWeb、Wappalyzer 等，为实现对 Web 站点的快速识别，使用响应报文头部中易被修改的 Banner 作为 Web 服务器指纹和 HTML 数据中可被删除的关键字作为 Web 应用指纹.

为解决研究工作与实际应用之间存在的上述问题，本文通过黑盒测试方法选取与 HTTP 协议[7]实现有关的 Web 服务器指纹，通过源码审计方法选取与功能相关的 Web 应用指纹，构建 Web 指纹库和设计 Web 指纹识别算法. 在此基础上，设计并实现了 Web 指纹识别工具 WebEye. 本文主要贡献如下：

1)不同于已有研究，采用响应报文的头部域[7]顺序和状态码定义作为 Web 服务器指纹，既可防止 Banner 欺骗，也不必构造大量的 HTTP 请求.

2)采用与功能相关且不易被更改的结构特征、静态文件及 Cookie 名作为 Web 应用指纹，克服了以往只采用关键字指纹导致的对 Web 应用识别率较低的不足.

3)基于上述选取的 Web 指纹，提出一种有效的 Web 指纹识别方法，并在此基础上设计并实现了 Web 指纹识别工具 WebEye.

1 相关工作

目前存在部分针对 Web 服务器指纹的研究，然而更多是侧重于 Web 服务器指纹的检测和防御[8]. 文献[5]通过对多种 Web 服务器的响应报文的分析，指出不同类型或版本的 Web 服务器对 HTTP 协议的实现不同，体现在对特定的 HTTP 请求的响应在词法、语法和语义等方面存在一些差异. 例如在响应报文的语法方面，Apache 的 Date 字段在 Server 字段的前面，而 Nginx 则相反. 文献[9]介绍利用 Banner 信息、HTTP 响应特征和特殊 HTTP 请求实现 Web 服务器识别：Banner 是最简单有效的方法，但是容易被修改，所以不能保证 Web 服务器识别的可靠性；HTTP 响应特征可避免 Banner 欺骗，但是需要同时与大量的其他特征一起才能准确推断 Web 服务器；而特殊 HTTP 请求依赖于不同服务器对相同 HTTP 请求的响应不同，这种方法能准确识别 Web 服务器类型，但是不能识别 Web 服务器的版本或版本范围. 文献[10]的设计思想是利用 Web 服务器行为的变化能推断出其类型及版本，通过对互联网上随机抽取的 Web 服务器对大量特殊 HTTP 请求的响应码的分析比较，提取其行为特征并对这些数据进行训练分类，从而实现对 Web 服务器类型的预测. 以上研究主要通过发送大量正常或畸形的 HTTP 请求，并分析 Web 服务器的响应报文，可准确识别 Web 服务器类型，但是其执行效率较低，而实际应用中主要使用最简单有效的 Banner 信息.

目前针对 Web 应用识别主要是通过 HTML 源码关键字、特殊文件及路径等实现[11]. HTML 源码关键字是提取标签 < meta name = “generator” content = “*application*” / > 的 content 值，是最快速的识别方法，但是该标签与 Web 应用功能无关，可以被修改或删除；特殊文件及路径是一些开源 Web 应用程序存在特殊的结构设计，如 WordPress 特有的文件路径/wp-includes/ 等. 文献[6]实现了一个情报搜集工具 W3 - Scrape，可用于对 Web 服务器、操作系统和 CMS 应用的识别，使用正则表达式和模式匹配对 HTML 源码关键字和 URL 的审查实现对 CMS 类型 Web 应用的识别. 文献[12]设计了一个 Web 应用版本识别工具 BlindElephant，通过将某种类型 Web 应用的特征文件与各个版本的文件进行比较从而实现对 Web 应用版本的准确识别，该工具的缺点是只能识别 Web 应用的版本，而不

能识别 Web 应用的类型. 文献[13]提出一个基于黑盒测试原理的 Web 应用识别方法, 搜集已知 Web 应用的特征信息包括 URL 模式、表单特征和关键字, 并建立指纹库, 将目标 Web 应用的特征信息与该指纹库进行比较从而实现 Web 应用的识别. 该方法有一定的局限性, 即不能修改 Web 应用的默认配置, 并且需要处理大量的 URL 以及对 HTML 数据的分析. 以上研究主要通过对大量 HTML 数据进行特征分析, 而实际应用中主要使用识别最快的 HTML 源码关键字.

综上所述, 目前针对 Web 服务器及应用指纹的研究与实际应用在执行效率和识别率上还存在一定的差距, 因此本文提出一种既能提高识别率, 也能提升执行速度的 Web 指纹识别方法.

2　Web 指纹的选取思路

2.1　Web 服务器指纹的选取思路

目前针对 Web 服务器指纹识别的研究主要通过发送大量精心构造的 HTTP 请求实现对 Web 服务器的准确识别, 但其识别速度较慢并且可能被网络安全设备当作恶意流量拦截; 而主流工具主要利用 Banner 快速获取 Web 服务器信息, 然而由于 Banner 易被修改而导致识别结果不可靠.

为解决 Web 服务器指纹识别的研究与实际应用之间存在的上述问题, 并进一步提高 Web 服务器指纹识别的准确率和效率, 本文以主流 Web 服务器[14], 即 Apache、Microsoft IIS 和 Nginx 的所有版本为研究对象, 以其对 HTTP 协议实现为研究内容, 以黑盒测试为研究方法, 通过向 Web 服务器发送以下 6 种 HTTP 请求, 并根据其响应报文的特征来选取 Web 服务器指纹.

- HEAD / HTTP/1.0
- PUT / HTTP/1.0
- DELETE / HTTP/1.0
- CONNECT / HTTP/1.0
- HEAD / JUNK/1.0 (异常协议类型)
- HEAD / HTTP/3.0 (异常协议版本)

通过对所有响应报文的研究分析, 选取以下 2 个能快速有效识别的 Web 服务器指纹.

1) 头部域顺序指纹, 见表 1. 主流 Web 服务器类型可由表 1 中的 4 个头部域的相对顺序准确识别. 例如, Apache 和 Nginx 对 HEAD / HTTP/1.0 请求的响应报文如下:

Apache 响应报文头部域相对顺序如下:

HTTP/1. 1 200 OK
Date: Tue, 07 Jul 2015 02:12:27 GMT
Server: Apache/1.3.33 (Win32)
……
Connection: close
Content-Type: text/html
……

Nginx 响应报文头部域相对顺序如下:

HTTP/1. 1 200 OK
Server: nginx/1.6.3
Date: Mon, 06 Jul 2015 03:03:10 GMT
Content-Type: text/html
……
Connection: close
……

表 1　头部域顺序指纹

Table 1　Fingerprints of fields ordering

类型	顺序
Apache	Date、Server、Connection、Content-Type
Nginx	Server、Date、Content-Type、Connection
IIS/5.0	Server、Date、Connection、Content-Type
IIS/6.0 – 8.5	Content-Type、Server、Date、Connection

2) 状态码定义指纹, 见表 2. 对于已知 Web 服务器类型, 其版本可由特定 HTTP 请求的响应状态码定义准确识别.

基于 Web 服务器对 HTTP 协议实现的研究分析进而选取的 Web 服务器指纹, 我们设计了一个既能防止 Banner 欺骗, 也不必构造大量 HTTP 请求的 Web 服务器识别算法.

表 2　状态码定义指纹

Table 2　Fingerprints of status code definitions

HTTP 请求	状态码定义	版本
JUNK 协议	400 Bad Request	Apache 1.3.x
	200 OK	Apache 2.x
PUT 方法	411 Length Required	Nginx 0.7.69 – 1.2.9
	405 Not Allowed	Nginx 1.4.7 – 1.9.2
DELETE 方法	501 Not Implemented	IIS 6.0
	405 Method Not Allowed	IIS 7.0 – 8.5

2.2　Web 应用指纹的选取思路

目前针对 Web 应用指纹识别的研究工作主要通过对大量 HTML 数据的分析来识别 Web 应用, 包括 HTML 源码关键字和特殊文件及路径, 由于需要处理较多 HTML 数据和文件, 降低了其执行速度; 而主流工具主要利用 HTML 源码关键

字实现 Web 应用的快速识别，但是由于关键字易被删除而导致识别率低.

为解决 Web 应用指纹识别的研究工作与实际应用中存在的上述问题，并提高 Web 应用的识别率，本文以主流的开源 Web 应用[15]为研究对象，以源码审计为研究方法，通过对 Web 应用的静态文件、源码以及结构设计的详细分析，并从以下 4 个方面选取 Web 应用指纹.

1)结构特征. Web 应用的类型可由其特殊的结构设计准确识别，该指纹只需从 HTML 数据的头部获取. 例如，如果 Web 应用的文件路径包含/wp-includes/，则该 Web 应用是 Wordpress.

2)静态文件. Web 应用的类型及版本可由其未被修改而被直接使用的静态文件准确识别. 例如，用于 Web 应用前端布局的文件 style. css，通过与目标 style. css 比较可准确识别 Web 应用的类型及版本.

3)Cookie 设计. Web 应用的类型可由开发者为其设计特殊的 Cookie 名准确识别. 例如，如果 Cookie 名为 django，则该 Web 应用是 Django-CMS.

4)关键字. HTML 源码关键字是主流工具选取的 Web 应用指纹，存在与功能无关且易被删除的缺点，但可作为 Web 应用的补充指纹. 通过对关键字在 HTML 源码中的分析，选取以下 4 个关键字指纹：

· < meta name = " generator " content = "*application*" / >

·Powered by *application*

· < meta name = "author" content = "*author*" / >

· <meta name = "copyright" content = "*copyright*" / >

其中，结构特征、静态文件和 Cookie 设计这 3 类指纹都与 Web 应用功能相关，有不易被修改或删除的特点，我们设计并构建了包含这 4 类 Web 应用指纹的 Web 指纹库，只需按照 Web 指纹库定义的格式向其中增加新类型 Web 指纹，即可实现对新增 Web 应用的识别，对 Web 应用的识别具有良好的可扩展性.

3 Web 指纹识别的设计与实现

3.1 Web 服务器识别算法设计

根据选取的 Web 服务器指纹的特征，设计一个简单有效的 Web 服务器指纹识别算法. 该识别算法分为 2 步：首先，发送 GET 请求，利用头部域指纹可准确识别 Web 服务器类型；然后，根据 Web 服务器类型构造特定的 HTTP 请求，利用状态码定义指纹可准确识别 Web 服务器版本. 该算法流程如图 1 所示，该算法的伪代码描述如图 2 所示.

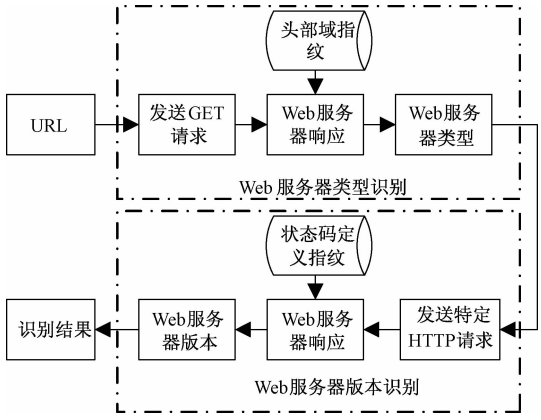

图 1 Web 服务器指纹识别算法流程

Fig. 1 Process of identifying web server

定义集合 ServerType:$T=\{t_1,t_2,\cdots,t_n\}$, 表示Web服务器类型; ServerVersion: $V=\{v_1,v_2,\cdots,v_n\}$, 其中$v_i=\{v_{i1},v_{i2},\cdots,v_{ik}\}$示第$i$类Web服务器的版本; Request:$R=\{r_1,r_2,\cdots,r_n\}$, 表示特殊的HTTP请求;OrderFingerprint:$O=\{o_1,o_2,\cdots,o_n\}$, 表示Web服务器头部域顺序指纹;CodeFingerprint:$C=\{c_1,c_2,\cdots,c_n\}$, 表示Web服务器状态码定义指纹.

```
Process WebServerIdentify
{
    req=findMethod(); //从集合R中选择HTTP请求
    file=sendRequest(req);
    order=getFieldsOrder(file); //获得头部域顺序
    for i=1 to n
        if(order==o_i)
            type=t_i;//Web服务器类型
            break;
    req=findMethod(type);
    file=sendRequest(req);
    code=getStatusCode(file);
    for j=1 to k
        if(code==c_j)
            version =v_ij;//Web服务器版本
            break;
    return type||version; //返回类型及版本
}
```

图 2 Web 服务器指纹识别算法伪代码

Fig. 2 Pseudo code of algorithm identifying web server

3.2 Web 应用识别算法设计

根据选取的 Web 应用指纹的特征，设计了一个有效的 Web 应用指纹识别算法. 该算法流程如图 3 所示.

识别一个 Web 应用需要以下 3 个步骤：

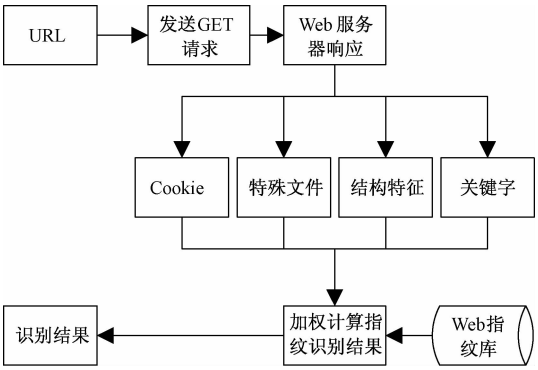

图 3　Web 应用指纹识别算法流程

Fig. 3　Process of identifying web application

1)URL 预处理. 对给定的 URL 进行规范化处理,分离 URL 中的主机名和资源路径,以便构造相应的 HTTP 请求.

2)获取 Web 应用指纹信息. 定义如下保存 Web 应用指纹的结构体变量 Fingerprint,根据对 URL 的预处理构造对应资源的 HTTP 请求,从响应报文中提取出所有的指纹信息并保存到以下 Fingerprint 结构体中.

```
typedef struct Fingerprint {
    char specialpath[20][255]; //结构特征集合
    char staticfile[10][32]; //静态文件 MD5 值集合
    char cookie[255]; //头部域的 cookie 名
    char keywords[4][1024]; //网页源码中关键字集合
    };
```

3)识别 Web 应用. 定义集合 Fingerprint: $F=\{f_1, f_2, \cdots, f_n\}$,表示 Web 应用的指纹;集合 Weight: $W=\{w_1, w_2, \cdots, w_n\}$,表示对应集合 F 中指纹所占权重,则识别结果为 $\max\{f_1\cdot w_1, f_2\cdot w_2, \cdots, f_n\cdot w_n\}$,表示指纹与权重乘积最大值的指纹对应的 Web 应用.

3.3　WebEye 的设计实现

根据前面选取的 Web 指纹及设计的识别算法,设计并实现了 Web 指纹识别工具 WebEye,其整体架构如图 4 所示.

在功能设计上,WebEye 设计为 2 个主要的功能模块,即 Web 服务器指纹识别和 Web 应用指纹识别.

在执行效率上,WebEye 在 Linux 平台下使用 C 语言多线程编程实现. 多线程并发执行使得 WebEye 可用于对海量 Web 站点快速准确地识别,并能根据运行环境指定线程数量,使其适用于各种硬件环境.

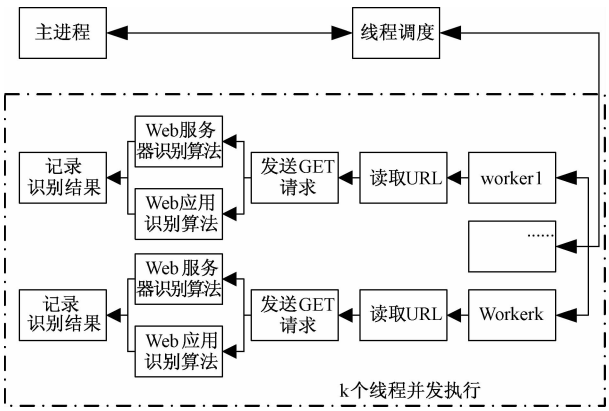

图 4　WebEye 的整体实现架构

Fig. 4　The overall implementation architecture of WebEye

4　实验

4.1　测试数据及环境

ZoomEye[1] 是一个针对网络空间的搜索引擎,根据其对 Web 服务的统计分析,将排名前 10 并且占统计量 97.439% 以上的 Web 应用作为 WebEye 的测试目标,具体 Web 应用及其部分指纹信息见表 3. 测试数据来源于这 10 种 Web 应用官方站点所展示的 Web 站点,并根据其比例一共选取 500 个测试 Web 站点. 这些 Web 站点既能作为衡量 Web 应用识别结果的标准,也能反映开发者对 Web 应用的修改特征,同时能检验选取的 Web 应用指纹的有效性. 实验环境是一台 PC 台式机(Ubuntu 15.04 系统,CPU 为 Intel Core i7 - 4790,内存 4 GB).

4.2　实验评估

表 4 描述了各 Web 指纹识别工具的实验结果. 其中,Web 服务器和 Web 应用分别表示其识别数量;时间表示识别过程所使用的分钟数,由于 ChromeSniffer 是浏览器扩展且未提供 API 接口,只能通过浏览器访问 Web 站点获得识别结果,所以未统计其时间. 如表 4 所示,WebEye 与其他 4 种工具相比:1)对 Web 服务器的识别率相差不大,但 ChromeSniffer、Wappalyzer 和 WhatWeb 使用 Banner 作为 Web 服务器指纹,因此其识别结果不完全正确;2)WebEye 对 Web 应用的识别率最高;3)WebEye 指纹识别执行时间最短.

表 3 测试目标 Web 应用及其部分指纹

Table 3 Target web applications and part of their fingerprints for testing

序号	Web 应用	关键字	结构特征	静态文件 MD5 值	Cookie 名称
1	wordpress	wordpress	/wp-admin/	53a151ba1af3acdefe16fbbdad937ee4	wp-settings
2	drupal	drupal	\	e6a9dc66179d8c9f34288b16a02f987e	\
3	discus!	Comsenz Inc.	\	c028c4822428e83a358c60a93ef65381	\
4	joomla	joomla	/joomla/	8894791e84f5cafebd47311d14a3703c	\
5	zblog	Zblogger	/zb_users/	9007f2a1385ce470b15e9c7b206c26ce	\
6	phpcms	Phpcms	/phpcms/	18fb0c67f6a7e5c7ad62fc37c5ab7637	\
7	phpwind	phpwind	\	cfc440185d836a969827f0fd52d38e03	\
8	TYPO3	TYPO3	/typo3 * /	8718c2998236c796896b725f264092ee	fe_typo_user
9	Django-Cms	Django	\	657120bc3388e4898fe9f996c1ab7af0	django
10	SonSpring	Nathan Smith	/sonspring/	f57b56c90dd8d71c4ed4e09d36d949fb	\

表 4 实验结果

Table 4 Experimental results

序号	工具	Web 服务器	Web 应用	时间/min
1	Httprint	463	\	2.14
2	ChromeSniffer	459	52	\
3	WhatWeb	468	52	2.09
4	Wappalyzer	423	238	54.55
5	**WebEye**	**476**	**313**	**1.10**

表 5 将 WebEye 与其他主流工具的 Web 指纹进行详细对比. Httprint 的指纹与 HTTP 协议实现相关，然而其 Web 指纹库已不再更新[16]，不适用于当前 Web 服务器的识别；ChromeSniffer 等浏览器扩展以及 WhatWeb 仅使用易被修改或删除的 Banner 及关键字作为 Web 指纹；Wappalyzer 选用 Banner 和关键字及结构特征作为 Web 指纹；WebEye 选取可防止 Banner 欺骗且不易被修改的 Web 服务器指纹和不易被删除且与功能相关的 Web 应用指纹.

表 5 Web 指纹识别工具的指纹对比

Table 5 Comparison of fingerprints selected using different web fingerprint identification tools

序号	工具	Web 服务器指纹			Web 应用指纹			
		头部域顺序	状态码定义	Banner	关键字	静态文件	结构特征	Cookie 名
1	Httprint	√	√	×	×	×	×	×
2	ChromeSniffer	×	×	√	√	×	×	×
3	WhatWeb	×	×	√	√	×	×	×
4	Wappalyzer	×	×	√	√	×	√	×
5	WebEye	√	√	√	√	√	√	√

表 6 描述各 Web 指纹识别工具在实验中起决定作用的 Web 指纹的分布情况，WebEye 与其他 3 种工具相比：1）Web 服务器识别的准确率提高 17.8%；2）Web 应用的识别率与执行效率最快的工具 WhatWeb 相比提高 52.2%；3）Web 应用的识别率与识别率最高的工具 Wappalyzer 相比提高 15.0%，且 Wappalyzer 执行速度较慢. 同时表 5 和表 6 也表明了本文选取的 Web 指纹的有效性.

表 6 Web 指纹比较

Table 6 Comparison of web fingerprints

类别	工具	指纹	识别数量	识别率/%
Web 服务器	Wappalyzer 等	Banner	387	77.4
	WebEye	协议实现	476	**95.2**
Web 应用	Wappalyzer 等	关键字	52	47.6
		结构特征	186	
	WebEye	结构特征	223	**62.6**
		静态文件	25	
		Cookie	13	
		关键字	52	

5 讨论

与已有的研究工作和主流工具的 Web 指纹识别原理相比，本文的研究工作有以下优势：

1）简化 Web 服务器识别. 从 Web 服务器对 HTTP 协议实现的角度进行研究，选取主流 Web 服务器的不易修改的指纹，并根据指纹特征设计了识别算法，既可防止 Banner 欺骗，也不必构造

大量的 HTTP 请求.

2)提高 Web 应用识别率. 从 Web 应用的源码、静态文件及结构设计的角度进行研究,选取与功能相关且不易被修改的 Web 指纹,并设计了识别算法,克服了关键字指纹的不足,提高了 Web 应用被识别的概率.

3)执行效率高. 对 Web 服务器的指纹识别不需发送大量 HTTP 请求即可实现准确识别,并缩小了 Web 应用指纹匹配的范围. 此外,WebEye 使用多线程并发执行,其执行效率优于 Web 应用识别率最高的工具 Wappalyzer.

下一步,我们预计克服前期由于人工处理和主观认知而导致静态文件指纹没有被充分利用的问题,通过对 Web 应用的静态文件进行更加深入的分析,提高 Web 应用的识别率. 更进一步实现一个基于 Web 指纹识别的渗透测试框架,利用 WebEye 的识别结果,自动查询漏洞库中该 Web 站点使用的 Web 服务器及应用的已知漏洞,并对这些可疑漏洞进行验证,进而生成详细的安全评估报告.

6　结束语

本文对主流 Web 服务器及应用进行深入分析,提出一种有效的 Web 指纹识别方法,解决了研究工作和实际应用之间存在的识别速度和识别率的问题. 通过选取不易被修改的 Web 服务器指纹和不易被删除的 Web 应用指纹,构建 Web 指纹库,设计 Web 指纹识别算法,进而实现了 Web 指纹识别工具 WebEye. 实验结果表明,WebEye 的 Web 指纹识别效果优于同类主流工具,并且使用多线程并发执行,可用于对海量 URL 的快速准确识别. 此外,WebEye 使用的 Web 指纹库具有良好的可扩展性,实现对新增 Web 服务器及应用的指纹识别.

参考文献

[1] ZoomEye. ZoomEye 网络空间搜索引擎[EB/OL]. [2016-01-20]. https://www.zoomeye.org/.

[2] Watson D. The evolution of web application attacks[J]. Network Security, 2007(11): 7-12.

[3] Goethem T V, Chen P, Nikiforakis N, et al. Large-scale security analysis of the web: challenges and findings[J]. Lecture Notes in Computer Science, 2014, 8564:110-126.

[4] Dukes L S, Yuan X, Akowuah F. A case study on web application security testing with tools and manual testing[C]// Southeastcon, 2013 Proceedings of IEEE. IEEE, 2013: 1-6.

[5] Lee D, Rowe J, Ko C, et al. Detecting and defending against Web-server fingerprinting[C]// Proceedings of Computer Security Applications Conference, 2002. 18th Annual. IEEE, 2002: 321-330.

[6] Karthik R, Kamath S. W3-Scrape-A windows based reconnaissance tool for web application fingerprinting[R]. arXiv:1306.6839.

[7] Fielding R, Gettys J, Mogul J, et al. Hypertext transfer protocol—HTTP/1.1[EB/OL]. (1999)[2016-01-20]. http://www.rfc-base.org/txt/rfc-2616.txt.

[8] Yang K, Hu L, Zhang N, et al. Improving the Defence against Web Server Fingerprinting by Eliminating Compliance Variation[C]// Proceedings of the 2010 Fifth International Conference on Frontier of Computer Science and Technology. IEEE Computer Society, 2010:227-232.

[9] Huang Z, Xia C, Sun B, et al. Analyzing and summarizing the web server detection technology based on HTTP[C]// Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on. IEEE, 2015: 1 042-1 045.

[10] Book T, Witick M, Wallach D S. Automated generation of web server fingerprints[R]. arXiv:1305.0245.

[11] Muller A, Meucci M, Keary E, et al. OWASP testing guide 4.0[EB/OL]. (2014)[2016-01-20]. https://www.owasp.org/ images/1/ 19/OTGv4.pdf.

[12] Thomas P. BlindElephant: Web application fingerprinter & vulnerability inferencing[EB/OL]. (2010-07-28)[2016-01-20]. https://media.blackhat.com/bh-us-10/presentations/Thomas/ BlackHat-USA-2010-Thomas-Blind Elephant-Web App-Fingerprinting-slides.pdf.

[13] Kozina M, Golub M, Groš S. A method for identifying Web applications[J]. International Journal of Information Security, 2009, 8(6):455-467.

[14] netcraft. Web server survey[EB/OL]. (2015-08-30)[2016-01-20]. http://news.netcraft.com/archives/2015/08/13/august-2015-Web-server-survey.html.

[15] 开源中国社区. 建站系统开源软件[EB/OL]. [2016-01-20]. http://www.oschina.net/project/tag/256/web-system.

[16] Net-square. Httprint signatures file[EB/OL]. [2016-01-20]. http://www.net-square.com/signature.txt.