浏览器指纹技术的研究与应用

杨立鹏,王 拓,樊春美

(中国铁道科学研究院,北京 100081)

摘 要:为了防止用户的信息被盗用,能够及时地发现用户的不正常登录,可以通过研究用户使用的浏览器特征,建立唯一标识用户的指纹信息。由于基于信息熵的浏览器指纹计算算法,会将发生细微变化的同一指纹生成不同的指纹,因此增加了指纹识别的错误率。为了改进该方法,介绍了基于浏览器指纹信息熵和相似度计算的浏览器指纹识别算法,即将不同的浏览器特征通过信息熵计算公式得到一个 ID,当出现同一用户而信息熵不一致时,根据相似度的计算进行判断。该算法综合考虑了指纹完全匹配和部分匹配的情况,提高了同一指纹的辨别能力。实验结果表明,该算法可以更好地识别用户。

关键词:浏览器指纹;设备指纹;JavaScript;Web 安全

中图分类号:TP39 **文献标识码**:A **文章编号**:1673-629X(2018)03-0169-05

doi:10.3969/j.issn.1673-629X.2018.03.036

Research and Application of Browser Fingerprint Technology

YANG Li-peng, WANG Tuo, FAN Chun-mei

(China Academy of Railway Sciences, Beijing 100081, China)

Abstract: In order to prevent the user's information from being stolen and find the abnormal login of users, we can study the user's browser features and establish an unique ID that identifies the user's fingerprint. As the browser fingerprint algorithm based on the information entropy will generate a different fingerprint for the same one with subtle changes, the error rate of fingerprint recognition is increased. In order to improve above method, we introduce the browser fingerprint identification algorithm based on the information entropy of the browser fingerprint and the similarity calculation. It gets an ID by information entropy calculation formula for different browser features. When the same users occur but their information entropy don't match, it is determined according to the similarity of the calculation. The proposed algorithm comprehensively considers the perfect matching and partial matching for fingerprint, which improves the discrimination to the same fingerprint. The experiment shows that it can be better identification of the users.

Key words: browser fingerprint; device fingerprint; JavaScript; Web security

0 引 言

随着互联网+的大力推广,电子商务平台也得到了飞速发展,但是这种新的交易方式具有虚拟性、即时性、跨行业地域性等特点,加上目前电子商务的法律法规,行业规则制定,以及第三方支付机制的不完善,安全保障措施不足、电子信息难以作为证据等导致电子商务领域的交易存在一定的风险。由于网络的开放性,交易者彼此的身份不确定,身份冒用的情况时有发生;在网购过程中,消费者面临的不是实物而是商家的描述,可能存在商家销售虚拟货物,彼此的信用体系并不完善;在互联网支付的过程中往往通过第三方支付平台来完成,消费者的身份信息和支付账户密码等信息容易遭遇木马、黑客的拦截,导致信息外泄。技术欺诈、信息欺诈、网络钓鱼、IP 欺诈等,这些都是目前电商领域最常见的欺诈行为[1-3]。

传统的反欺诈手段一般是通过账户、银行卡、手机、IP 等维度进行身份确认。对近年来各大电商网站反欺诈技术手段的应用情况进行汇总研究发现,随着技术的发展,这些维度逐渐变得易于篡改,反欺诈效果十分有限,通过这些维度难以识别真正的电商客户。验证的方式最初使用用户名、密码等,后来升级为使用动态口令令牌、数字证书等,增强了安全性,但是目前最好的方式是利用生物特征的唯一标识来进行认证,通过用户是谁来解决识别真实用户的问题[4-6]。

浏览器指纹技术是指采集电商客户发起交易的设备环境信息,通过逻辑运算,为每一台设备上的浏览器

收稿日期:2017-02-15 **修回日期**:2017-06-22 **网络出版时间**:2017-11-15

基金项目:中国铁路总公司 2016 年重点项目(2016X004-G);中国铁路总公司 2015 年重点项目(2015X003-A)

作者简介:杨立鹏(1982-),男,博士研究生,研究方向为网络安全。

分配一个唯一的编号。这个编号综合了用户的多种信息,并不是单纯地依赖用户名和密码,提高了用户的识别能力,并且通过这个唯一标识,反欺诈技术人员可以观察到客户交易环境、交易行为的变化,提高反欺诈的能力。

1 浏览器指纹关键算法和特性研究

浏览器是电商用户使用最多的程序,通过确定浏览器的指纹发现电子商务中潜在的问题具有非常重要的意义。如果根据每个用户浏览器的版本、使用浏览器的操作习惯及用户的常见设置,为其建立唯一的指纹,并且保持一定时间范围内的稳定性,那么分析者就可以根据浏览器指纹来确定具体用户的身份了。

1.1 浏览器指纹算法研究

只要用户访问网站,就会泄漏一些信息。这些信息不仅包括比较私密的信息,如用户 IP 地址、口令、密码,还有经常使用的浏览器的型号、版本和配置信息等等,所有的信息都会泄漏用户的行为。为了更好地表示用户,把所有的信息综合起来,形成浏览器的指纹信息。因此采用基于浏览器信息熵的方法唯一地标识使用者的身份。

1.1.1 浏览器指纹信息熵

信息熵具有非常多的应用,这里通过建立指纹的信息熵来表示指纹信息。将每个浏览器的特征转换成离散数值,为提高计算效率,避免使用字符串。假定指纹生成函数为 $F()$,每输入一个新的浏览器信息 x,都会得到一个新的浏览器指纹 $F(x)$,该函数会满足一个离散概率密度函数 $P(f_n)$,$n \in [0,2,\cdots,N]$。浏览器指纹的结果往往会受到使用者、软件开发人员、技术事故等多种因素的影响,并且特征集合的大小和采取的数值也不能确定。为了避免引发过多问题,建议 N 值不要取得太大。使用式(1)表示函数输出结果的自信息量[7-9]:

$$I(F(x)=f_n)=-\log_2(P(f_n)) \tag{1}$$

这里采用 2 作为对数的底数,自信息量 I 表示为占用的 bit 位数。$P(f_n)$ 分布的信息熵为所有浏览器自信息量的期望值,表示为:

$$H(F)=-\sum_{n=0}^{N}P(f_n)\log2(P(f_n)) \tag{2}$$

由于浏览器存在多个组件,针对每个组件定义它的信息量及信息熵:

$$I_s(f_{n,s})=-\log_2(P(f_{n,s})) \tag{3}$$

$$H_s(F_s)=-\sum_{n=0}^{N}P(f_{n,s})\log_2(P(f_{n,s})) \tag{4}$$

对于指纹的组件不独立的情况,应该采用条件概率,将式(3)改为:

$$I_{s+t}(f_{n,s},f_{n,t})=-\log_2(P(f_{n,s}\mid f_{n,t})) \tag{5}$$

在浏览器的指纹特征确定后,对于每一个登录的用户,使用上述公式即可得到一个浏览器的指纹特征。这个值唯一标识一个用户,但是即使是同一个用户也不能保证他的配置是一成不变的,因此浏览器指纹算法应该能够具有识别指纹变更但同属一原指纹的能力,这里采用加强指纹算法。

1.1.2 基于信息熵和相似度计算的加强指纹算法

浏览器加强识别指纹算法是专门解决来自同一个浏览器但指纹不相同的情况,当信息熵不一致时,进行相似度计算。指纹变化的原因可能是用户每次访问的时间不同,或者由于需要查看浏览信息的需要,对浏览器做了一些必要的更改。在这种情况下,浏览器指纹算法应该能够识别这种异常,对这种差异有一定的容忍度。加强指纹算法就是用来解决这个问题的,两次指纹信息,如果大部分指纹的信息是一致的,那么就应该认为这是一个原指纹的变种,而不能认定为一个新的指纹。

加强指纹采用相似度的计算来实现[10]。对于每个浏览器特征,采用二进制的形式进行区分,简化相似度的计算方法。对于浏览器指纹的每个特征项分别进行完全匹配,如果符合则为 1,否则为 0。对于不同的特征项给予不同的权重,因此指纹的相似度计算如式(6):

$$S=\sum_{n=1}^{N}E_i\alpha_i \tag{6}$$

其中,S 表示采集的指纹与数据库指纹的相似度;E_i 表示获取的浏览器指纹特征项;α_i 表示 E_i 特征的权重大小。

设置相似度的阈值 M,假如 S 的值大于 M,则认为该指纹是加强指纹,否则认为是新的指纹。该算法综合考虑了各种因素对于浏览器指纹的影响,具有非常强的可靠性,而且该算法实现起来比较简单,因此是一种优秀的加强指纹判定算法。

1.1.3 浏览器指纹算法实现

建立一个高效的用户指纹算法,需要解决的关键问题有两个:一个是使用的指纹识别特征,好的指纹识别特征是提高浏览器指纹识别算法有效性的关键因素;另外就是如何识别同一个指纹的不同变化,可以避免建立过多的指纹信息,提高识别同一指纹的准确性。对于识别同一个指纹的不同变化,使用加强指纹算法。综合多种影响因素建立的浏览器指纹算法[11-12]的实现过程如下所示:

(1)用户访问网站时,采集用户的用户名、登录时间以及 IP 地址信息,用来提高浏览器的真实性。

(2)采集浏览器的基本信息,并通过配置的关联

性,确定浏览器各个特征。

(3)使用 1.1.1 介绍的算法,针对浏览器的不同组件,生成浏览器对应的信息熵。

(4)将用户名与信息熵建立对应的关系,并将浏览器的特征加到对应的后面。

(5)用户登录进来后,计算该指纹的信息熵,检查指纹库中是否存在该用户的指纹,如果没有,则将该用户的指纹信息加到指纹库中,否则对比信息熵是否一致。

(6)如果该用户的信息熵一致,则是回访用户,否则,使用式(6)计算登录用户与指纹库中指纹特征的相似度。

(7)如果相似度在一定的容忍范围内,则判断为回访用户,更新对应的指纹信息,否则为该用户建立新的指纹。

1.2　多变性与稳定性的分析

在生成浏览器指纹时,既要为满足所有用户的需求生成足够多的设备指纹,又要能够对同一个用户的微小变化生成同一个指纹 ID。浏览器指纹的生成受多种因素的影响,如用户更改使用的操作系统的版本,修改浏览器的配置,更改浏览器的版本等等,每个不同的配置都会产生不同的信息熵,从而生成不同的设备指纹。选择的影响因素越细,生成的设备指纹的种类就越多,这样就能够满足有着细微差别的用户,也能够拥有不同的设备指纹。但是即使是同一个用户,他的设备指纹也会随着一系列事件的变化而发生改变,比如安装或者删除插件,改变字体,更改 Cookie 的配置。为了维护用户设置指纹的稳定性,不能因用户做一些改变就又生成不同的设备指纹,为此要求采用的指纹特征不能过于详细。因此在指纹的特征选择时,要同时考虑生成指纹的多变性和稳定性,这也同时决定着指纹生成算法的有效性。

1.3　加强指纹算法的验证

利用 Java 语言开发的网站,在用户登录时,添加收集浏览器信息和生成指纹的算法,为了对比加强指纹的识别能力,分别实现了基于信息熵的指纹识别算法和基于信息熵和相似度的指纹识别算法。使用不同的用户名,修改不同的配置,进行实验 200 次,部分实验结果如表 1 所示。

通过计算识别的结果可得,识别指纹的错误率降低了 10%,从而证明了加强指纹具有比较好的效果。

2　浏览器指纹识别系统的设计

在正常情况下,当用户通过浏览器访问一个网站时,该网站便会在使用者电脑上留下 Cookie 痕迹,根据这个用户信息,可以追踪用户的线上行为,但是用户可以将 Cookie 信息删除,导致无法查看用户的行为。但是浏览器指纹能够正常地跟踪用户。因为它不在用户的计算机上留下持久的痕迹,具有非常隐秘的性质。

表 1　对比实验结果

序号	用户名	修改项	修改详情	信息熵判定结果	加强指纹判定结果
1	HJKac	更改 Cookie 的级别	Cookie 级别由中改到低	N	Y
2	Song_0208	更换 IP 地址	192. 168. 10. 1 →192. 168. 6. 25	N	N
3	HJKac	无	无	Y	Y
4	KK_10715	更换浏览器	IE→火狐	N	N
5	Yang_M	更改 SSL 状态	清空 SSL 状态	N	Y
6	Li_Yong_bo	Plugins 变化	增加了 IE 缓存工具	N	N
7	Wang_guang	操作系统版本	Win8→Win10	N	N
8	Yin_chao	无	无	Y	Y
9	Xfan	更改字体	仿宋→宋体	N	Y

注:N 为不同指纹,Y 为同一个指纹。

如何标记设备一直是在研究的问题,目前最常用的就是 Cookie 或者 Session。Session 具有一定的时效性,不能验证每次登陆的是否为同一用户。使用 Cookie 就需要通过浏览器 set-cookie 存储标识到客户端,但是会受到条件的限制。首先浏览器有定时清除的功能,用户也可以手动清除、禁止写入,导致这种标识的稳定性很低,另外标识的粒度比较复杂,不能适配到所有的场景。

因此现在需要一种基于多维度特征的综合标记方法,能从浏览器中获得 UserAgent, Language, Color Depth, CPU class, Platform, Canvas fingerprinting, WebGL fingerprinting 等多项维度的数据,采用浏览器指纹计算算法得到一个具有综合评价的唯一标识,可以更好地跟踪用户的行为。

2.1　建立浏览器指纹特征

用户只要访问网站就可能泄露信息,这些信息包括用户 IP、口令、密码等敏感信息,也包括一些不是特别敏感的信息,比如浏览器的型号、版本和配置信息等等。这些信息又可以分为唯一信息和非唯一信息。唯一信息指的是可以唯一标识一个设备的信息,非唯一信息指的是那些用户可以更改的信息。

为了获取这些信息,浏览器插件、Cookies、用户提交的 Http 请求和 javaScript 脚本被广泛用于收集浏览器特征。这些工具不仅能够收集独特的浏览器特征信息,还能收集浏览器的提供商、浏览器的版本号、使用的插件和字体等信息。但是这种获取方式有一定的限制,就是用户必须访问浏览器才能使用这些方法。目前一些用户会更改代理字段甚至伪造,甚至为了保护

访问的隐私,浏览器通常使用 https 的加密数据传输模式。但是浏览器的流量差异多是由于内核不同导致的,用户不同,相同浏览器的流量差异不会太大,因此采用流量分析法也是一种识别浏览器的有效方式[13]。

常用的浏览器指纹特征主要有 7 种:User Agent、HTTP_ACCEPT Headers、浏览器插件信息、系统字体库、显示器设置(屏幕信息)、时区信息、Cookie。表 2 展示了各种浏览器指纹特征对应的获取方式和特征描述。

表 2 主要的浏览器指纹特征

序号	指纹特征	特征获取方式	特征描述
1	User Agent	HTTP,JS	浏览器、硬件等信息,变动较多
2	HTTP_ACCEPT	HTTP	http 头信息,变动较少
3	Plugins	JS	插件信息,变动较多
4	System Fonts	JS	系统安装字体,变动较少
5	Screen	JS	显示器设置,稳定
6	Timezone	JS	时区信息,稳定
7	Cookies	JS	Cookie 设置,变动较少

在实际应用中,需要对以上 7 种浏览器指纹特征进行进一步完善、细化和区分。由于每个特征可能拥有不同的状态,所以使用不同的二进制位来表示每个组件的状态。

2.2 浏览器指纹的识别流程

浏览器指纹主要用来解决账号密码盗用的情况,能够及时发现交易者使用环境的变化,识别交易的真实用户,保障网上交易的安全性。浏览器指纹的识别流程主要分为浏览器特征的获取、浏览器指纹的生成、浏览器指纹的识别,具体流程如图 1 所示。

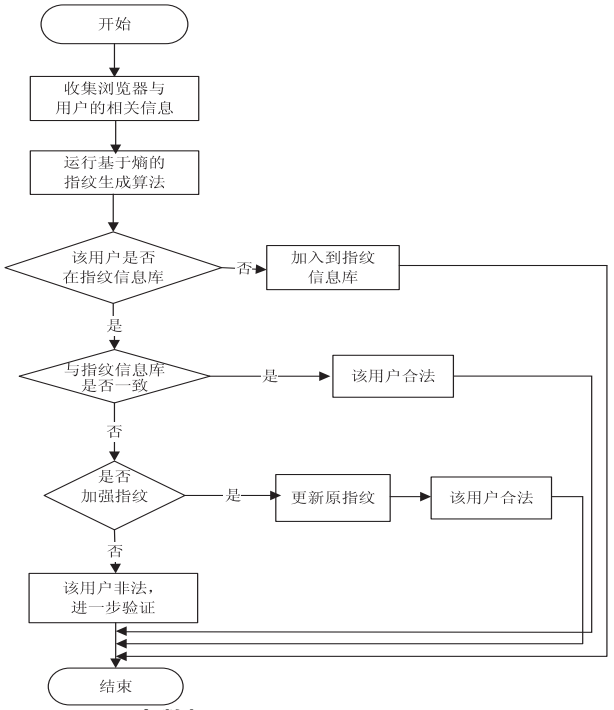

图 1 浏览器指纹识别流程

在服务器端,除了原服务系统,在相应的数据库中增加一个指纹信息表,当用户登录时,首先对其使用的浏览器生成一个浏览器指纹,然后判断这个用户是否存在,如果不存在说明是一个新的用户,将它的设备指纹加入到指纹信息表中。

确定该用户是注册过的用户后,判断他的浏览器指纹,看当次登录的环境是否与以前的一致,如果一致,说明该用户没有问题,可以正常地使用网站。否则,需要进一步判断是否对浏览的环境做了修改导致指纹不同,如果是,将数据库中的指纹更改为当前的浏览器指纹,否则,该用户每次登陆的环境差异过大,需要进一步对其进行验证。通过这个识别过程,可以及时地发现冒用者,为网络安全提供了保障。

2.3 浏览器指纹的技术架构设计

浏览器指纹用来核验真实性,在访问和交易的过程中,实时发现具有非常重要的意义,因此可以采用如图 2 所示的技术架构。这样的设计既能满足大量用户设备指纹存储的问题,也能实时地计算设备指纹和比对工作。

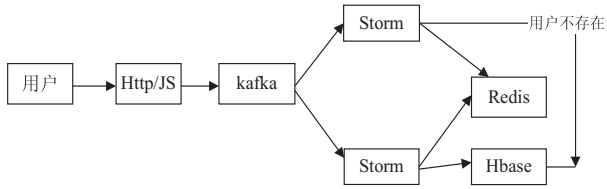

图 2 浏览器指纹的技术架构

(1)采集指纹。

Http/Js 用来收集设备指纹的信息,Http 与 Js 结合的方式,可以将设备指纹的信息收集得更加全面。

(2)浏览器指纹的计算。

图 2 中上面的 Storm 主要用来实现设备指纹的计算,判断设备指纹是否在 Redis 中,如果该用户不在浏览器指纹数据库中,则将用户的信息和浏览器指纹一起写到 Hbase 中,否则,判断该指纹是否与数据库中的指纹一致;如果一致,该用户合法,继续执行操作,否则,需要判断是否是加强指纹;如果是,更新 Hbase 的浏览器指纹,否则该用户有一定的风险,需进一步验证。

(3)浏览器指纹的存储。

Hbase 可以解决大数据量存储的问题[14],是对于访问量大的系统的最佳选择。使用 Storm 将数据库备份到 Redis 中,可以提高浏览器指纹比对的效率。Redis 作为内存数据库,查询效率非常高,可以迅速地得到比对结果。

(4)其他服务。

搭建 Kafka 集群,是建立一个分布式消息队列系统,将收集的信息通过后台发送给 Kafka,防止收集的

数据不能被及时处理,起到一个缓冲的作用。Kafka 有两个接收端,除了上面用来计算浏览器指纹的 Storm,还有下面的 Storm,在计算浏览器指纹的同时,它根据用户的信息,去 Hbase 中查找是否存在该用户的浏览器指纹,如果存在,则将信息同步到 Redis 中。

Storm 是分布式实时计算,具有较强的实时性,在这里用来计算浏览器指纹,具有较高的效率,能够保证系统的实时性,因此使用上面的技术架构可以达到非常好的应用效果。

3　结束语

随着网络技术的迅速发展,手机、平板电脑等各种智能设备的加入,网络访问和交易也变得越来越频繁,各种安全问题也逐渐凸显。传统的基于用户名密码的验证手段已经不能满足安全的需要,因此需要综合用户的多种信息为其建立一个唯一标识,以此来提高认证的准确性,保护用户的信息安全。

文中介绍了浏览器指纹的生成算法及其应用系统架构的设计。在指纹识别的过程中,综合使用了完全匹配和部分匹配的方式,可以有效地识别有效指纹的变种,降低错误增加指纹的数目,达到最佳的指纹识别效果。在浏览器指纹的架构设计中,综合考虑了高访问量和实时性的问题,在原系统的基础上,加上指纹识别系统,可以提高认证用户的安全性。随着电子商务遇到越来越多的安全问题,浏览器指纹在机器人防范、用户访问特征、客户端数据等方面必将有着更广泛的应用。

参考文献:

[1] 杨青龙. 网购中的欺诈行为研究[J]. 东西南北 · 教育观察,2011(9):19.

[2] MIYAZAKI A D,FERNANDEZ A. Consumer perceptions of privacy and security risks for online shopping[J]. Journal of Consumer Affairs,2001,35(1):27-44.

[3] 黎志成,刘枚莲. 电子商务环境下的消费者行为研究[J]. 中国管理科学,2002,10(6):88-91.

[4] 吉　绚,胡　曼,刘广宇. 网上购物安全性现状分析[J]. 中国水运:理论版,2006,4(12):116-117.

[5] 吴凌娇. 网上购物安全问题探讨[J]. 江苏技术师范学院学报,2006,12(4):68-73.

[6] BODA K,FÖLDES Á M,GULYÁS G G,et al. User tracking on the web via cross-browser fingerprinting[M]//Information security technology for applications. Berlin:Springer,2011:31-46.

[7] 张梦媛. 浏览器的安全访问及指纹识别技术[D]. 南京:南京邮电大学,2012.

[8] YEN T F,HUANG X,MONROSE F,et al. Browser fingerprinting from coarse traffic summaries:techniques and implications[C]//International conference on detection of intrusions and malware,and vulnerability assessment. [s. l.]:[s. n.],2009:157-175.

[9] 徐　晏,张代远. 基于浏览器的用户身份识别系统[J]. 计算机技术与发展,2013,23(8):79-82.

[10] 王　维,王伟平,王建新. 一种基于升级浏览器指纹的用户识别方法[C]//信息安全漏洞分析与风险评估大会. 出版地不详:中国信息安全测评中心,2014.

[11] MUÑOZ - GARCÍA Ó,MONTERRUBIO - MARTÍN J,GARCÍA-AUBERT D. Detecting browser fingerprint evolution for identifying unique users[J]. International Journal of Electronic Business,2012,10(2):120-141.

[12] 姚全珠,蒋鹏飞,颜丽菁,等. 基于浏览器指纹技术的预防黄牛党挂号系统[J]. 计算机应用,2016,36:276-279.

[13] 李周辉,黄燕群,唐　屹. 浏览器识别研究[J]. 信息网络安全,2016(3):34-39.

[14] 陆　婷,房　俊,乔彦克. 基于 HBase 的交通流数据实时存储系统[J]. 计算机应用,2015,35(1):103-107.

(上接第 168 页)

[6] 王俊杰,徐小刚,胡运发,等. 鱼眼投影在虚拟实景中的应用研究[J]. 小型微型计算机系统,2004,25(2):287-290.

[7] 芦鸿雁. 基于层次包围盒的碰撞检测算法研究[J]. 计算机与数字工程,2008,36(2):23-25.

[8] 李运锋,刘修国. 基于方向包围盒投影转换的轮廓线拼接算法[J]. 计算机应用,2011,31(12):3353-3356.

[9] CIRNE M V M,PEDRINI H. Marching cubes technique for volumetric visualization accelerated with graphics processing units[J]. Journal of the Brazilian Computer Society,2013,19(3):223-233.

[10] WATSON B,LUEBKE D. The ultimate display:where will all the pixels come from[J]. Computer,2005,38(8):54-61.

[11] 韩俊刚,蒋　林,杜慧敏,等. 一种图形加速器和着色器的体系结构[J]. 计算机辅助设计与图形学学报,2010,22(3):363-372.

[12] 黄　龙. 基于 GLSL 三维渲染效果的研究与应用[D]. 西安:西安科技大学,2014.

[13] WOO J H,SOHN J H,KIM H,et al. A 195 mW,9.1M vertices/s fully programmable 3-D graphics processor for low power mobile devices[J]. IEEE Journal of Solid-State Circuits,2008,43(11):2370-2380.

[14] 陈继选,王毅刚. 基于 OSG 的 GLSL 着色器编辑环境[J]. 计算机系统应用,2011,20(3):153-156.

[15] WOLF D. OpenGL4.0 shading language cookbook[M]. [s. l.]:PACKT Publishing,2011.