DOI: 10.14188/j.1671-8836.2017.05.002

移动端 Web 浏览器 HTTP 流量注入的监控与屏蔽

谢梦非[1,2], 傅建明[1,2,3†], 王应军[1,2], 彭国军[1,2]

(1. 武汉大学 计算机学院, 湖北 武汉 430072;
2. 武汉大学 空天信息安全与可信计算教育部重点实验室, 湖北 武汉 430072;
3. 武汉大学 软件工程国家重点实验室, 湖北 武汉 430072)

摘 要: 提出一种在服务器部署前端脚本程序的监控方案, 对移动端 HTTP 流量注入行为进行监控. 基于监控数据, 对注入内容、注入主体进行分析. 分析结果显示, 超过 4%的移动端 Web 客户端会话在传输过程中被篡改, 这样的篡改包括注入普通广告、注入恶意广告、网络运营商增值服务、恶意代码、虚假访问代码、页面重定向等, 注入主体和网络运营商、地域、网络环境有关. 基于此, 提出了 4 项针对这些注入的屏蔽方案, 包括在服务器部署 HTTPS、CSP、部署检测脚本与 HTTPS 以及在客户端部署访问限制程序等方案, 并对这些方案进行测试. 测试结果表明, 在服务器部署 CSP 的方案成本低且准确率较高; 针对高性能客户端, 在客户端部署访问限制程序的方案能有效屏蔽流量注入.

关 键 词: 网络安全; HTTP 流量注入; 页面篡改; 页面变化检测

中图分类号: TP 309.5 **文献标识码**: A **文章编号**: 1671-8836(2017)05-0385-12

Monitoring and Blocking Methods of HTTP Traffic Injection in Mobile Web Browser

XIE Mengfei[1,2], FU Jianming[1,2,3†], WANG Yingjun[1,2], PENG Guojun[1,2]

(1. School of Computer, Wuhan University, Wuhan 430072, Hubei, China;
2. Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan University, Wuhan 430072, Hubei, China;
3. State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, Hubei, China)

Abstract: This paper first presents a method of deploying the front-end scripts in the server to monitor the HTTP traffic injection of the mobile browsers. The analysis of the injection contents and the injection entities based on the monitoring data suggest that over 4% mobile devices' sessions are modified during transmission. These modifications include the injection of the advertising, the injection of malvertising, the injection of the ISP value-added services, the injection of the malicious code, the injection aiming at improve false access and the injection aiming at page redirecting. The injection entities are found related to the network operators, the regions and the network environments. This paper also presents 4 methods including deploying HTTPS in the server, deploying CSP in the server, deploying the detection scripts and HTTPS in the server and deploying the access restriction procedure in the client, aiming at blocking the injection. The test results of these methods show that the method of deploying CSP has low cost and high accuracy and the method of deploying the access restriction procedure in the client is effective for the high performance client.

Key words: network security; HTTP traffic injection; page modification; page changes detection

收稿日期: 2017-01-10 **†通信联系人** E-mail: jmfu@whu.edu.cn
基金项目: 国家自然科学基金资助项目(61373168, U1636107)
作者简介: 谢梦非, 男, 硕士生, 主要研究方向为 Web 安全. E-mail: xmf1992@qq.com

0 引 言

互联网上广泛存在的HTTP流量注入严重影响用户上网体验,并带来安全隐患.近年来随着移动互联网的发展,移动终端设备更加普及.这些移动终端设备大多数处于4G、3G、2G或Wi-Fi网络环境下,往往更易受到攻击.用户从浏览器发出的请求到Web服务器之间的路径上会经过防火墙、路由器、入侵防护设备、缓存设备等,该路径上的任何设备都能修改网络流量内容,也能注入虚假的网络流量.这种修改或注入流量的行为通常简称流量注入.流量注入早期被用于DNS、HTTP错误消息的提示,后来被广泛用于广告、缓存与安全审查.同时,流量注入也被攻击者用于注入攻击向量,如恶意代码.流量注入是目前广受关注的热门话题,过去的几年国内外有很多关于流量注入的报道[1~4].

文献[5~8]对流量注入的细节进行了研究.目前有两类研究方向:以服务器为中心的研究方向和以客户端为中心的研究方向.以服务器为中心的研究采用在服务器上部署监控脚本程序的方案,这类方案的监控对象是指定的一台或多台服务器;以客户端为中心的研究则是监控一个区域内的一个或多个客户端,由于这些客户端访问网页的不确定性,监控的服务器数量也是不确定的.文献[5~7]是以服务器为中心的研究,文献[8]是以客户端为中心的研究.

Reis等[5]提出并实现了一个web tripwire通用脚本程序,该脚本程序可以被嵌入在任意网页上,监控并上报传输中的网页的变化.在测试过程中,他们上线了一个嵌有web tripwire脚本程序的网页,并吸引志愿者去访问.他们的研究结果显示,超过1%的客户端Web页面在传输过程中被篡改,大多数的篡改来自于安装在客户端上的软件,还有一些篡改是因为网络运营商压缩流量,此外有5家提供免费Wi-Fi热点的组织向用户的网页中注入广告.

Zimmerman[6]利用在线广告交易的漏洞,利用Flash广告可以调用JavaScript程序的特性来监控页面内容.他们以DSP(广告需求平台)客户的身份,评估了4家DSP的漏洞利用情况,并选择了其中一家DSP部署广告位.测试结果表明,部分广告位发生了脚本注入、meta标签被篡改和页面中链接被替换的情况.

Weaver等[7]的研究采用了ICSI Netalyzr[8]工具,该工具可将他们开发的Java程序部署在网页上.他们开发的Java程序可用于执行一系列测试,比如爬取指定服务器的数据.测试结果中也发现了客户端网页被Web代理篡改的情况,这些代理的功能包括缓存修改、编码、404响应、网关登录、注入内容等.

Nakily等[9]的研究采用了以客户端为中心的研究方式.研究者在三所大学和一所企业的网关上部署了监控设备,用来监控这些机构的网络流量.经过分析和追踪,发现主要有两类大型商业网络运营商存在带外注入的行为,这类带外注入表现得非常隐蔽,在一定时间内是不可重现的.该文献提出了两个防止带外注入的方法,包括分析IP包的identification和TTL值的方法,并对后者进行了加强.这是首次针对商业网络运营商的TCP带外注入的研究.

本文采用了以服务器为中心的研究方式,分析对象是移动端Web浏览器.采用在网页中部署脚本的方式对注入数据进行监控,并对注入数据中的注入内容与注入主体进行分析.本文提出了4项针对流量注入的屏蔽方案,包括部署在客户端上的方案和部署在服务器上的方案,并对这些方案的性能、易普及程度、成本、误报率和漏报率进行分析和测试.

1 带内注入与带外注入

HTTP流量注入包括带内注入与带外注入两种方法.

带内注入是一种典型的中间人攻击,实施这种攻击的中间人是网络路径上的一个节点,客户端与服务器之间的TCP包经过这样的中间节点,则中间节点可以对这种TCP包进行监控和修改.由于HTTP协议是明文的,中间人对传输中的HTTP包可以进行修改与替换.这个修改替换过程就是带内注入.

与带内注入不同的是,带外注入攻击者可以是客户端与服务器之间的节点,也可以是节点旁的旁路设备.注入主体监控客户端与服务器之间的流量,并在合适的时间伪造一个数据包发送至客户端,这个时间是在服务器向客户端发送真正的数据包之前.根据TCP协议,客户端收到伪造的TCP包之后,会忽略掉之后服务器向客户端发送过来的真正的数据包.这个过程就是带外注入.带外注入有两个特点:第一,带外注入中的客户端会接收到两个数据包,且在接收到第一个数据包之后,会忽略第二个数据包;第二,和带内注入相比,带外注入主体直接向客户端发送一个经过伪造的数据包,这个数据包不

是基于原数据包修改而成的,而是直接伪造的,这对攻击设备性能要求较低.

无论是带内注入还是带外注入都要求数据是明文的,比如 HTTP 数据.使用 SSL 加密过的数据是无法被篡改的.带内注入与带外注入的过程可用图 1表示.

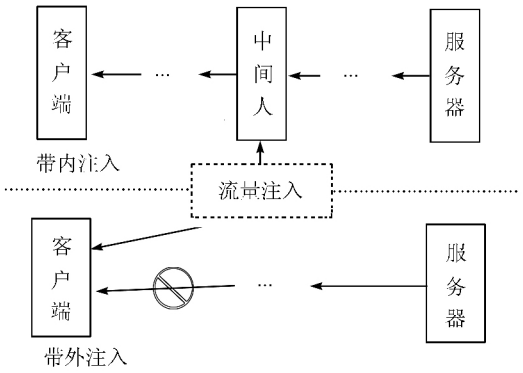

图 1　HTTP 流量注入

Fig. 1　HTTP traffic injection

2　监控方案

2.1　监控数据收集

本文的测试平台是一个移动终端访问量较大的网站.为了监控到终端设备的注入情况,需在该网站上部署一个前端脚本程序,该程序检测并上报浏览器端的网页信息.首先建立一个基于网站地址的白名单,并将该网站的资源域名添加至该白名单,以保证具有这些域名信息的资源不被脚本识别为站外资源.部署的脚本程序分为三个部分,一部分用于检测是否有站外资源,也就是白名单以外的资源;一部分用于在已经检测到站外资源的情况下,检测页面的变化;还有一部分是将变化前后的网页文档和站外资源列表一起上传到服务器.

浏览器访问部署该脚本程序的网页后会下载这个脚本程序并执行,接着脚本开始依次检测页面上的每个标签的 src 属性值(即资源 URL)是否存在于白名单中,若存在白名单以外的资源 URL,则将该页面标记为受感染页面,并第一次上传当前的站外资源 URL 列表和当前的页面文档.需要注意的是,这里的页面文档并不像文献[5]那样使用 document.documentElement.innerHTML,因为他们没有考虑到网页被嵌入在框架中的情况,且没有考虑到〈html〉标签本身的属性的变化.这里使用 window.top.document.documentElement.outerHTML,可以确保即使网页被嵌在〈iframe〉标签中,程序仍然能获取到全部的网页文档.接着脚本开始检测网页上的变化,这种变化通常伴随着新的标签加载.检测方法是每隔一段时间检测页面文档的 Hash 值,若某一次的 Hash 值和上一次检测到的 Hash 值不相等,则上报当前的站外资源 URL 列表和当前的页面文档,直到浏览器跳转到其他页面或页面文档的 Hash 值不再变化.

需要注意的是,每一次上传过程中,除了需要上传站外资源 URL 列表和当前页面文档之外,还需要上传用户 ID、当前 URL、父页面 URL(如果父页面存在)、用户代理、网络环境(2G、3G、4G 还是 Wi-Fi)、IP 地址和上报时间等关键信息.网络环境的获取方法有三种:对于有一些移动端浏览器的用户代理字段已经包含网络类型,如部分 QQ 浏览器和所有微信内置浏览器,直接从用户代理中提取;第二种,从浏览器提供的 navigator.connection[10] 中提取,目前移动端 Android 浏览器(版本号不小于 2.2)、移动端 Firefox 浏览器都支持该 API;第三种,如果前面两种方法不可行,则可通过网速测试的方法估算网络类型.

监控程序共运行 22 d,这 22 d 内产生的页面总会话数量达 2 976 395 次,监控程序收集的数据达 238 172 条.

2.2　将监控数据分至会话组

注意到在上述数据收集过程中,一个被感染的页面可能产生多次数据上传.事实上大多数被感染页面有不低于 2 次的数据上传.为了方便地处理这些数据,需要将同一个页面下的所有数据生成一个会话数据.本文的数据整理程序首先将数据集基于用户编号进行排序,以保证同一个用户的数据是连续的.接着程序开始遍历每一行的用户数据,对会话方式进行判断.判断会话的方式有两种:一种是当 URL 发生变化,或者若同一个 URL 停留时间超过 1 min,则视为用户离开或跳转到其他页面,即进入下一个会话;另一种是判断用户的 ID 值是否发生变化.根据这两种判断方式将用户进行分组.最终,238 172条原始数据被分进 124 535 个会话.

此外,在分组的同时还需要建立方便每个会话查询的站外资源 URL 列表.这样可以方便地获取注入内容,还可以对会话进一步分组.最后收集到的列表包括 1 433 个站外资源 URL.我们去掉了这些 URL 的参数,避免出现大量重复 URL.

除了对浏览器端的地理位置、网络运营商等特性进行整理,还需要使用 IP 地址库将收集到的 IP 地址映射到各省市的地域和网络运营商.

2.3 获取注入内容

为了方便地分析这些被注入的站外资源的内容,需要将获取到的站外URL下载.最终有1 110个URL被正常下载.将1 433个URL进行两次初始检测,包括恶意网址检测(通过MWSL-hosts[11])和广告网址检测(通过EasyList和EasyListChina[12]),共检测出9个恶意URL和213个广告URL,其中9个恶意URL在后来的分析中被标记为普通广告.我们将已下载的样本在VirusTotal[13]上进行扫描,进一步确认是否存在恶意脚本.在VirusTotal提供的53个病毒库中,仅有Dr. Web[14]的病毒库将其中4个样本识别为病毒,我们将这4个样本进行分析,发现只是存在一些有安全风险的JavaScript函数,比如eval、unescape等,实际上并没有造成安全风险,因此将其标记为普通广告.这1 110个可下载样本,加上初始检测结果,共计1 145个样本.对于这些注入内容的分析见3.1节.

2.4 将会话分至资源组

每个会话对应一个站外资源的集合.当收集到足够多的会话数量时,可以考虑将具有相同或相似资源的会话分进同一个资源组,这样,每个资源组可以代表一种注入方式,从而实现对注入主体的分析.本文采用以下方法确定一个具有相同或相似资源的会话:对于两个会话,如果其中一个会话的资源组包含另一个资源的资源组,则认为这两个会话属于同一组.

本文的程序对每一个会话进行遍历,并采用上述方法确定该会话的资源是否属于某一个资源组,如果该会话的资源不属于任何一个资源组,则将该会话的资源创建为一个新的资源组.最终得到了196个资源组.

需要注意的是:第一,只需对每个会话中的第一次上报的资源进行提取,因为第一次上报的资源往往是起始资源,后面的资源通常是由这些起始资源加载的;第二,需要对比的两个资源组应该是基于主机地址而不是基于完整URL的,因为一部分URL对应同一个主机地址;第三,在统计资源组的同时,还应当统计每个资源组对应的会话个数,因为之后需要将资源组按会话个数进行排序,需要记录产生资源个数最多的会话,这类会话包含最多的资源个数,有助于更加全面的分析.基于这些资源组的注入主体分析见3.2节.

3 监控结果分析

依上节所述方法,监控程序运行22 d收集到的238 172条原始数据被分进了124 535个会话,这些会话占总会话数量的4.2%.因此,超过4%的移动端Web客户端会话在传输过程中被篡改,具体分析如下.

3.1 注入内容分析

依次对1 145个注入样本进行评估,并将他们进行分组.分组基于以下两点:注入样本(可能是图片、脚本、网页或其他内容)的内容,以及被注入的网页与原始网页的区别.由于数据量较大,本文建立了一个人工审核系统,功能包括显示每个注入样本的内容和对应的被注入的网页的内容.基于这两点对注入样本进行分组标记,分组结果如表1所示.

普通广告 绝大多数注入的内容是普通广告.广告内容包括购物、理财、社交APP、游戏APP和其他一些常见的广告内容类型.根据广告的呈现方式将这些广告进行分类.最常见的是,广告脚本在浏览器界面下方固定一个广告横幅,用户一旦点击这个横幅,则跳转到广告商提供的页面.另外有部分广告被嵌入在网页内容中,包括一些新闻资讯,这会占用大量的网页空间,导致原始内容无法正常显示.有部分广告是一个浮层,它会遮住大多数网页内容,并让用户决定是否关闭它以查看正常网页内容.有部分广告是一个悬浮窗,它提供一个悬浮的图标,用户一旦点击这个图标,浏览器便进行跳转,这类图标大多

表1 一部分已知样本注入分类统计

Table 1 The stats of known injection samples

注入类型	样本种类数	受影响话数	受影响用户数
普通广告	809	70 250	17 996
恶意广告	38	435	215
恶意代码	3	14	6
网络运营商增值服务	254	9 954	3 959
提高虚假访问量	17	11 766	2 876
页面重定向	1	70	22
其他	23	3 199	1 745

数是一个返回主页的图标,达到诱导用户点击的目的.

注意到一些广告的展示和原始网页内容有关系.有 10 例广告脚本将一些政府网站的域名加入白名单,如果脚本检测到网站域名属于这个白名单,则不进行广告展示.有 1 例广告脚本检测网页是否包含尺寸较大的图片(图片宽度不低于 300 px,高度不低于 150 px),只有存在这样的图片,才进行广告展示;有 1 例广告脚本只针对 qq.com 等特定的 243 个域名进行注入广告,即便这样的脚本仍然被注入到网页中.

一部分广告的展示与用户代理、IP 地址、网络环境有关.广告脚本可以通过用户代理获取用户的浏览器和操作系统,通过 IP 地址获取用户的地理位置.有 5 例广告脚本针对北京、上海、广州、深圳、杭州、南京等地投放与其他地方不同的广告;有 6 例广告样本基于用户的浏览器和操作系统展示不同的 APP 下载界面;有 2 例广告样本,在使用国外 IP 访问时,提示广告不在该国展示,而在国内访问时广告却正常显示.

一些广告是在 Wi-Fi 环境下被注入的.对一家酒店的广告脚本的分析结果表明,在用户连上该酒店的 Wi-Fi 后,注入设备会向网页底部注入一个广告,这个广告指向该酒店的门户网站.在另一例 Wi-Fi 环境下样本中,可看到一则和网络带宽提速相关的悬浮广告,这则广告指向一家智能路由器生产厂商的商城.

本文发现一些广告存在的安全隐患,特别是在 Wi-Fi 环境下注入的广告.有 2 例广告脚本泄露了用户和 Wi-Fi 设备的 MAC 地址,这可能使网站运营者获取到用户的 MAC 地址,造成一定安全隐患.

恶意广告　根据广告内容和广告行为判定一个广告是否是恶意广告.从广告内容看,一些恶意广告携带一些诱导用户点击的信息,但往往不会提供用户想要的结果.比如诱导用户点击拆开一个红包,但实际是下载理财产品 APP,再比如诱导用户获得免费的影视资源,但实际上是下载窃取隐私的 APP.若用户不慎点开了这样的广告,并按照广告的要求进行操作,可能会导致隐私泄露或账户被盗.还有一些恶意广告传播违法信息.从广告行为看,一些恶意广告有遮住或替换原始网页的行为,有 1 例恶意广告样本展示了一个弹窗浮层,但这个弹窗浮层是无法关闭的,如果用户不关闭浏览器,则不得不点击这类广告.正常情况下,广告只会上传它本身的 Cookie 提供给广告交易平台进行分析,但本文在 1 例恶意广告脚本中发现了上传全部 Cookie 的情况,这将导致用户账户信息的泄露.

恶意代码　有 3 个样本(不含恶意广告样本)被发现存在恶意代码.其中 2 例恶意代码脚本的程序将用户的账户信息进行上传,这些信息包括该网页的 Cookie 信息、IP 地址、地理位置和网页 URL,其中 Cookie 信息的泄露会导致用户的登录状态被泄露,造成安全隐患.另外 1 例脚本在原网页上添加一个带有滑动验证码的浮层,这遮住了原始网页的大部分内容,这个浮层告诉用户只有滑动验证码验证成功才能继续访问,但是无论用户怎样滑动滑块,都无法验证成功,基于这一现象可推测这一滑动验证码可能接入了一个验证码识别系统,利用巨大的流量,达到快速人工识别验证码的目的.

网络运营商增值服务　有 254 个样本被我们识别为网络运营商增值服务,这类服务的出现频次仅次于普通广告.这是在移动互联网环境下出现的新型注入形式,这类注入由网络运营商主导,用于向用户展示流量相关内容,比如本月剩余流量,同时向用户提供流量充值入口,或者推荐流量包套餐.这类服务大多是以一个悬浮窗的方式默认显示在正常网页的右下角,也有一些嵌在网页内容中或展示在网页底部.不同地域的网络运营商提供的服务呈现方式不相同,同一网络运营商在不同地域呈现方式也不一定相同.对这些网络运营商增值服务样本的统计结果见表 2.其中运营商 A 注入的地区数量和展示方式远超过其他两个运营商.

需要注意的是,网络运营商往往有获取用户手机号的权限.某地运营商 C 的 2 个样本中被发现了经过 DES 加密的手机号,但这并不存在安全风险,除非 DES 密钥泄露.本文以 mobileNum、phone-

表 2　网络运营商增值服务样本统计

Table 2　The stats of samples of the ISP value-added services

网络运营商	样本数量	地区数量	默认展示位置	备注
运营商 A	190	10	网页右下角或底部	相同展示位置样式相同,不同展示位置样式不同
运营商 B	7	1	网页底部	
运营商 C	57	2	网页左下角	两个地区样式不相同

Num、mobileNo、phoneNo等作为关键词在会话中搜索,同样也没有发现类似的泄露.对所有这类样本发起的请求进行分析,也没有找到获取明文手机号的入口,但这不意味着没有类似的隐患,这可能和时效性有关.

提高虚假访问量 17个旨在提高虚假访问量的样本并不注入实际内容,而是注入〈img〉、〈iframe〉等标签,用于加载一些网页.注入者的目的可能是利用这些流量带来虚假的访问数字.大多数提高虚假访问量的样本嵌入一段统计代码,单纯地增大统计数字.另外一些样本则有所不同.在1个样本中,脚本加载了一个高度与宽度都是1px的〈iframe〉标签.在3个样本中,脚本加载了搜索引擎对某一关键词的搜索页面,这可能会提升该关键词在搜索引擎中的权重.还有1个样本只在微信浏览器进行注入,这可能是因为需求方需要提高网站从微信端访问的比例数字.此外,某网页内的5个样本,使用〈img〉〈iframe〉等标签,指向苹果应用商店和一家直播网站.

页面重定向 只有1个样本被发现存在页面自动重定向代码.可能是因为在页面自动跳转之前,没有足够的时间来上报数据.但这个样本说明了这一现象的存在,这个样本的代码将浏览器直接跳转到一个微信图文页面,这个图文页面提供一个连载小说的内容,达到引导用户关注微信公众号或付费查看小说的目的.

其他 其他样本主要是上述样本加载所需要的公共JavaScript库和1个上传数据的样本.这个上传数据的样本处于Wi-Fi环境下,向服务器上报本机MAC地址和Wi-Fi热点的MAC地址,这可能造成MAC地址被泄露的安全隐患.

3.2 注入主体分析

每个资源组对应一种注入方式,但每种注入方式可能对应一个或多个资源组.本文根据注入的内容和资源组所在的主机地址依次对196个资源组进行分析.如果这些资源组中包括一些相同主机的注入行为,则将这些资源组分到一个注入组,这样的注入组对应同一个注入主体.这需要确定一个资源组的资源注入链条的起始资源,也就是这个资源组最开始注入的资源是哪些.虽然这些资源组采用的是一个会话中首次上报的数据中的资源,但这并不意味着这些资源一定就是被注入的初始资源,它们也可能是被初始资源加载的其他资源,而这些资源在首次上报之前就被加载.为了确定初始的注入资源有哪些,需要提取出属于同一个资源组的所有会话组所对应的用户,对这些用户在同一时间段内,同一网络条件的其他会话中的资源进行提取,得到这些会话的首次上报资源.由于这样的首次上报资源的数据量较大,这些首次上报资源中往往会包含初始资源.按照这些资源出现的次数由大到小进行排序,提取出现次数最多的一个或多个资源.采用上述方法对资源组进行遍历,大多数情况下,如果从两个资源组采用上述方法提取到资源是相同的,则这两个资源组属于同一个注入组.少数情况下,我们将上述方法提取出的出现次数第二的资源作为判定方法,因为在9例特殊情况中,出现次数最多的资源脚本在触发其他资源加载的操作完成之后,将它自己从网页文档中移除.除此之外,我们还将这些注入资源中具有相同URL参数的注入资源判断为同一个注入主体,这类资源的注入方式可能来自于同一种注入系统,比如一些网络运营商增值服务的注入系统.有一家主体的注入系统一部分采用IP地址,另一部分采用域名,这在我们进行IP地址反查后才被发现.将监控数据划分至注入组的整个分析过程可用图2表示.

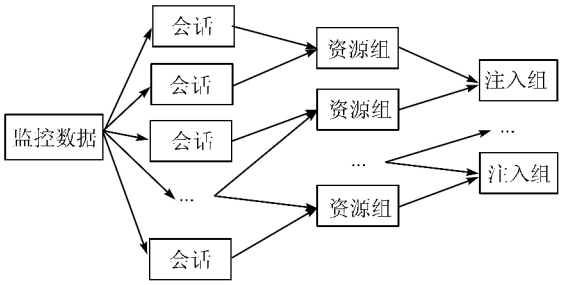

图2 将监控数据划分至注入组

Fig. 2 Grouping the initial data into the injection groups

对注入组的命名是根据主机地址或者实施主体名称来进行的.另外,对这些注入主体影响的会话数量、注入目的、主要域名或主机地址、注入地域、网络类型、ICP备案主体进行了统计.最终,我们得到了61个注入组.表3列出了前21个影响超过100个会话的注入组和5个特别的注入组,以及这些注入组的属性.

根据注入目的,这61个注入组包括3个提供网络运营商增值服务的主体、2个提高虚假访问量的主体、2个注入恶意广告的主体,2个注入恶意代码的字体、1个页面重定向主体和51个注入普通广告的主体.通过注入的资源主机地址可以查到这些地址对应的备案信息,这可能与注入主体有一定关联.基于这些会话用户的IP地址和用户代理,对出现频次较高的被注入地域以及网络环境进行统计,统计

表 3　影响超过 100 个会话的注入组和一些特别的注入组

Table 3　The injection groups effecting over 100 sessions and some special injection groups

序号	注入主体名称	影响会话数量	注入目的	主要域名或主机地址	注入地域	影响线路	影响网络环境	ICP 备案信息
1	liuzhi520	36 876	广告	liuzhi520.com yhzm.cc	不限	不限	不限	某网络公司
2	运营商 C	34 098	网络运营商增值服务	117.131.*.* 218.205.*.* 等 9 个	9 个市	运营商 C	不限	无
3	hsgsrj	14 154	广告	hsgsrj.com	不限	不限	不限	个人
4	运营商 A	11 453	网络运营商增值服务	113.57.*.* 212.180.*.* 等 14 个	19 个市	运商营 A	不限	无
5	baldui	10 279	广告	baldui.com st.wayayaya.com	不限	不限	不限	某网络公司
6	ximeifang	7 748	广告	ximeifang.com 139.129.*.*	不限	不限	不限	某网络公司
7	51.la	993	广告	51.la	不限	不限	不限	企业
8	myaigou	836	广告	myaigou.com	2 个省	运营商 A	Wi-Fi	某网络公司
9	yuanyiyi	688	不详	yuanyiyi.com	1 个省	运营商 A	Wi-Fi	个人
10	221.179	559	广告	221.179.*.*	1 个市	运营商 C	4G/3G/2G	无
11	s9w	469	广告	s9w.cc	不限	不限	Wi-Fi	个人
12	jiajv	295	恶意广告	jiajv.net	不限	运营商 B	不限	无
13	sybspools	267	广告	sybspools.com wdzsb.com.cn	不限	不限	Wi-Fi	某研究所
14	aimeiren	247	广告	aimeiren.top	不限	运营商 C	不限	无
15	minisplat	242	广告	minisplat.cn	不限	不限	Wi-Fi	个人
16	yiqisee	234	广告	yiqisee.cn	1 个省内的 2 个市	运营商 A	4G/3G/2G	某贸易公司
17	reekr	183	广告	116.62.*.*	不限	运营商 A	不限	无
18	86str	136	广告	86str.com	不限	运营商 C	不限	个人
19	iuni	121	广告	iuni.com.cn	1 个省	运营商 A	不限	无
20	219.148	117	广告	219.148.*.*	1 个省	运营商 B	不限	无
21	138138138.top	106	广告	138138138.top	不限	运营商 A	Wi-Fi	无
				⋮				
29	运营商 B	39	网络运营商增值服务	219.146.*.* 10.234.*.*	2 个市	运营商 B	不限	无
33	dotinapp	35	提高虚假访问量	dotinapp.com showself.com lnk0.com onelink.me cqxmgs.com	不限	不限	Wi-Fi	含 2 家网络公司
40	hantinghotels	23	广告	hantinghotels.com	不限	运营商 B	Wi-Fi	某酒店
52	wangfanwifi	7	广告	wangfanwifi.com	1 个市	运营商 A	Wi-Fi	某 Wi-Fi 服务提供商
60	witown	3	广告	witown.com	1 个市	不限	Wi-Fi	某网络设备制造商
				⋮				

结果表明有一些注入与用户所处的地域以及网络环境有关.

注意到,序号为2、4、29的注入组注入的是网络运营商增值服务,其中运营商C、运营商A的增值服务覆盖的会话和地域较多.这类增值服务往往对不同的地域采用不同的主机地址资源,比如运营商C对每一地域采用一个对应的主机地址,而运营商B则对19个地域注入了13个IP和1个域名.这类增值服务和用户所处的网络环境无关,无论是在4G、3G、2G还是在Wi-Fi环境下,只要使用运营商的线路,都有一定概率被运营商注入增值服务.

一些广告的注入可能和网络运营商有关.注入组8、9、16、17、19、21仅在运营商A的网络中进行注入,注入组12、20仅在运营商B的网络中进行注入,注入组2、10、14、18仅在运营商C的网络中进行注入.其中,注入组8影响的网络环境只有Wi-Fi,这可能和该运营商网络环境下,从Wi-Fi设备开始的某一个节点有关系,包括Wi-Fi设备、调制解调器、小区宽带设备等,这些节点不会使蜂窝移动网络受到影响.注入组8影响了2个不同的省,影响范围较大,因此和小区宽带设备无关.和注入组8类似的还有注入组9、21.注入组10、16可能和运营商基站设备有关,因为它没有影响到宽带线路.注入组10、12、14、17、18则对地域和网络环境没有限定,这说明这些注入可能和运营商主干线路上的网络设备有关.

一些广告的注入和运营商、地域、网络环境都没有关联.注入组1、3、5、6、7就属于这类情况.这里有两种情况:第一种情况与用户Wi-Fi设备到CDN[15](内容分发网络)的节点都没有关系,而是与CDN到网站服务器这条线路上的节点有关,这涉及到网站资源白名单内的一个或多个URL,这样的URL传输的内容可能被CDN到网站服务器上的一个或多个节点篡改;第二种情况是白名单的URL并不可信,因为我们网站的白名单中有一个第三方流量统计服务,它本身可能会加载额外的脚本,这也会导致其他使用其服务的网站的安全性受到影响.

一些广告的注入只和Wi-Fi有关系,这样的注入极有可能由用户的Wi-Fi设备发起.注入组11、13、15就属于这类情况.使用搜索引擎对注入组11的关键词进行检索,可发现这和最近流行的一款免费路由器有关系,这款路由器在系统设置中默认开启了"购物比价"功能,这会引起相关的注入.通过ICP备案查询和WHOIS查询的方法对注入组15进行追踪,发现注入主体和一家网络广告公司有关.

另外,有一些特别的注入组.注入组33注入的主要内容是为一些苹果应用商店的APP进行刷页面点击次数,这可能和目前流行的APP搜索优化业务有关.注入组40是一家连锁快捷酒店的Wi-Fi设备,它将连接该酒店Wi-Fi的用户的网页注入酒店的门户广告.注入组52、60提供公共Wi-Fi连接服务,同时向用户的网页注入广告,而他们的ICP备案指向两家提供Wi-Fi服务或制造网络设备的公司.

3.3 局限性

本文的监控方法无法检测到以下内容:

部分不加载检测脚本的注入.包括一部分直接绕过网页的注入,这些注入可能会将用户浏览器直接跳转至与网站无关的页面,导致前端脚本程序无法检测.比如,这些注入可能直接返回一个含3xx跳转[16]的HTTP响应头部,或者带有跳转功能的〈meta〉标签,或是带有页面跳转功能的JavaScript脚本.这类注入方式大多属于带外注入.但是,一些采用〈iframe〉标签对测试网页进行加载的带外注入,是可以被检测到注入存在的.尽管如此,无法完全确定这类注入方式是带内注入还是带外注入.从注入者的角度看,如果采取带外注入的方法进行〈iframe〉注入,则对实施注入行为的服务器性能要求较低,更节省成本且易于操作.

屏蔽本站脚本的注入.如果一个注入使网页的检测脚本无法正常运行,则网页的检测脚本无法检测或上报这样的注入.

无法访问的跨域注入资源.如果注入的资源是跨域的(一般是这样),且指定特定线路上的用户能访问,则无法获取到这些资源的具体内容,只能获取到这些资源的URL.同时,由于浏览器的跨域限制,网页内的检测脚本是无法获取到这些资源内容的.

对其他网站的注入.本文的监控方式是以服务器为中心的,服务器数量是有限的,因此本文的前端检测脚本无法监控到客户端访问其他网站的行为.

4 屏蔽方案

对于网站主体来说,需要向用户传递准确的内容,同时要保证服务器安全性、稳定性,以及成本的最小化.对于用户来说,要保证安全性,也希望能看到正确的内容,同时保证终端设备的流畅性.这两者都有屏蔽流量注入内容的需求.基于此,本文提出以下4项针对流量注入的屏蔽方案.

1) 在服务器部署HTTPS.由于HTTPS经过

加密传输,注入主体无法获取到明文数据.从客户端安全性和服务器安全性看,采用 HTTPS 会提高安全性.从服务器成本和性能看,采用 HTTPS 会提高成本和降低性能.从客户端性能上看,由于 HTTPS 有加密和解密的过程,因此会占用一部分 CPU 资源.从客户端部署难度上看,HTTPS 无须在客户端上进行部署.从服务器部署 HTTPS 的普及程度来看,目前国内 HTTPS 普及程度仅 25.8%,排名处于末尾[17].

2)在服务器部署 CSP[18].CSP(内容安全策略)提供一个白名单机制,限制只能请求白名单内指定的 URL.CSP 可以指定特定类型的资源加载,并可以限制这些资源的 URL.如果服务器有确定资源的白名单,则可以考虑采用 CSP 进行部署.服务器部署 CSP 后,即使网页被注入了资源,这些资源也不能被加载,因此可以屏蔽掉大多数广告注入.但是对于网站主体来说,如果网站接入了第三方广告平台的动态广告,则会造成动态广告资源无法正常加载,因为网站主体无法确定这些广告的来源.基于这一考虑,大多数接入第三方动态广告的网站主体不会采用 CSP.采用 CSP 方案的其他风险则在于一些带外注入导致浏览器跳转到第三方网站的情况,还有注入主体将 HTTP 响应头部的 CSP 进行篡改的情况.

3)在服务器部署检测脚本和 HTTPS.由于在服务器端部署 HTTPS 的方案对性能有所要求,可以考虑只在被注入的情况下使用 HTTPS 协议.这需要一个用来检测页面是否被注入站外资源的脚本程序.正常情况下用户仍使用 HTTP 协议访问网页,如果脚本检测到网页上的注入,则浏览器将页面跳转至相应的使用 HTTPS 协议的页面.由于被注入的会话数量占所有会话比例较低,HTTPS 在这部分的损耗较少.本文之前提到,部署 CSP 对于接入第三方动态广告的网站主体是不可行的,因为无法确定这些广告会加载哪些资源.但是网站主体对于在网页的哪一个位置加载广告是确定的,因此脚本可以忽略站内合法第三方广告的位置.当然,这不能排除注入脚本篡改第三方广告的内容的情况.此外,如果检测脚本无法被正常加载,则该方案无效.

4)在客户端部署访问限制程序.基于服务器的屏蔽方法的最大的一个缺陷在于即使某一个网站采用了这些屏蔽方案,对于终端用户来说,当他们访问其他的网站的时候,仍然有大量被 HTTP 注入的机会.但是,如果一些程序被部署在客户端上来防止 HTTP 注入,则终端是否被注入,不受访问的网站的主体的影响.与在服务器上部署不同的是,在客户端上无法部署白名单,因为在浏览器访问一个网站之前,往往并不提前知晓该网站的站内资源.但是基于上文的分析方法,可以总结出一个黑名单,并对黑名单进行限制访问.我们只需要将监控到的初始注入资源添加到这样一个黑名单中.这样的访问限制往往可以采用 PAC[19](代理自动配置)实现.这种方法的局域性在于目前只能从特定的网站中获取这样的注入资源,这样的黑名单不具有普遍性.

5　屏蔽方案测试

5.1　屏蔽方案测试方法

本节具体描述上述屏蔽方案所用的测试程序.

1)在服务器部署 HTTPS.本文现有的测试服务器采用的是阿里云 ECS 主机,该主机拥有 8 核 CPU、8 GB 内存、8 Mb/s 出网带宽.部署的前端环境是 nginx,并采用 CDN 加速.在 nginx 和 CDN 上都设置了 HTTPS 证书与私钥.采用阿里云的 cloudmonitor 插件对每分钟的 CPU 平均占用率、网络流入带宽等数据进行监控.采用 CPU 占用、带宽占用两个指标来考查 CPU 占用率和网络流入带宽.其中 CPU 占用指标为一段时间内每分钟 CPU 平均占用率之和除以这段时间内页面访问次数;带宽占用指标为一段时间内每分钟平均带宽之和除以这段时间内页面访问次数.

在网页上部署一个检测脚本,用于检测是否发生注入,同时上报页面文档与页面加载时间.对页面加载时间的获取需要用到浏览器的 performance.timing 接口[20].

该方案参照的原始数据是未采用 HTTPS 加密时的服务器的 CPU 占用指标、服务器带宽占用指标以及页面平均加载时间.

对于一个网页,如果检测脚本在 HTTPS 环境下检测到站外资源的注入,则认为发生一次漏报.由于发生 HTTPS 错误时,网页无法正常加载,导致检测脚本无法正常加载,因此无法检测误报率.

2)在服务器部署 CSP.我们现有的测试网站包含两个主要域名 main.alice.com 和 static.alice.com(域名已做匿名处理),其中 main.alice.com 用于呈现主网页,static.alice.com 用于加载 CSS、JavaScript 和图片等静态文件.网站还有一些图片是直接向〈img〉标签内的 src 设置 data 值来加载的.网站 AJAX 请求的 URL 只限定在 main.alice.com.网站内没有内嵌和内联 CSS,但是有内嵌

〈script〉标签,而且有使用 setTimeout 等函数,因此 script-src 字段需要用到'unsafe-eval'与'unsafe-inline'.

综合以上条件,设置 CSP 策略如下:

```
default-src 'none';
img-src data: static.alice.com;
script-src static.alice.com 'unsafe-eval' 'unsafe-inline';
style-src static.alice.com;
connect-src 'self';
report-uri //main.alice.com/api/set_csp_log;
```

网站设置了一个 report-uri,用于上传 CSP 错误.

该方案也需要记录 CPU 占用指标与带宽占用指标,并由检测脚本上报页面平均加载时间.该方案参照的原始数据是未采用 CSP 时的相关指标.

需要在页面中插入检测脚本,用来测试误报率与漏报率.对于一个网页,如果检测脚本发生上报但未出现 CSP 报错,则认为发生了漏报,如果检测脚本未发生上报而 CSP 发生报错,则认为发生了误报.

3) 在服务器部署检测脚本和 HTTPS.该方案需在服务器部署一个检测脚本,若该脚本检测到站外注入,则将网页跳转至相应的 HTTPS 页面.若未检测到站外注入,则对页面加载时间进行上报.同时,HTTPS 页面中也需要部署一个脚本,对页面加载时间进行上报.

由于该方案的检测脚本与本文检测误报及漏报的检测脚本对注入的检测方法是相同的,因此该方案没有检测误报与漏报的必要.

该方案参照的原始数据是未采用该方案时的服务器 CPU 占用指标、服务器带宽占用指标以及页面平均加载时间.

4) 在客户端部署访问限制程序.从上述 61 个注入组中提取出 247 个域名或主机 IP 作为主机地址黑名单,生成 PAC 文件并将该文件应用至测试环境.

由于该方案是基于客户端的,而目前没有足够多的客户端参与测试,因此可获得的样本数量较少.这里采用模拟注入环境的方法解决这一问题.构造一个注入环境.选取 20 类注入组,每组取 5 个页面,共计 100 个注入页面.按 4% 的注入率计算,共需 2 500个页面,因此未被注入的页面有 2 400 个.注入环境采用 AnyProxy 与 Proxifier 进行透明代理,模拟注入资源.

客户端采用型号为 Intel i7-6600U 的 CPU、512GB 固态硬盘、Windows 10 操作系统,并采用 64 位 Chrome 浏览器模拟 iPhone6 和 4G 环境(4 Mb/s).针对客户端环境开发了一个 Chrome 扩展,它将每个页面刷新 5 次,并记录测试页面平均加载时间.

对于一个被注入站外资源的样本,如果没有检测到访问限制程序的拦截,则认为该样本被漏报.对于一个未被注入站外资源的正常样本,如果检测到了访问限制程序的拦截,则认为该样本被误报.我们记录误报与漏报的样本编号,以及发生次数.这里采用 Chrome.webRequest 接口[21]判断是否有资源被访问限制程序拦截.

5.2 屏蔽方案测试结果

基于服务器的屏蔽方案需要采用的指标包括 CPU 占用指标的提升比例、带宽占用指标的提升比例、页面加载时间的提升比例、误报率、漏报率.基于客户端的屏蔽方案需要采用的指标包括客户端平均页面加载时间提升比例、被注入页面平均加载时间提升比例.对屏蔽方案的测试结果见表 4.

表 4 屏蔽方案测试结果

Table 4 The test results of the blocking methods

%

方案	服务器 CPU 占用提升比例	服务器带宽占用提升比例	客户端平均页面加载时间提升比例	被注入页面平均加载时间提升比例	误报率	漏报率
在服务器部署 HTTPS	37.80	37.10	25.90	N/A	N/A	0.009
在服务器部署 CSP	0.40	0.40	2.50	N/A	0	0
在服务器部署检测脚本和 HTTPS	1.80	1.70	5.80	N/A	N/A	N/A
在客户端部署访问限制程序	N/A	N/A	11.30	−89.90	0	0

从 CPU 占用提升和带宽提升比例来看,在服务器部署 HTTPS 的方案需要耗费较多的 CPU 资源和带宽,平均页面加载时间也随之增加.另外 2 种服务器方案则可以忽略不计.

在测试服务器上部署 HTTPS 之后,只有 0.009% 的网页被篡改,这个比例可以忽略不计.CPU 日平均占用率由 3.41% 提升到 4.70%,这带来约 37.8% 的性能损耗.平均单次访问流量则由 8.55 KB提升至 11.72KB,这带来 37.1% 的流量增长.采用 HTTPS 对于访问量大的网站来说需要更多的或性能更好的服务器支撑,网站运营成本随之提高.

在服务器部署 CSP 方案的误报率与漏报率均为 0. 最初,测试程序检测出很高的误报率,而经过进一步的人工检测,发现 CSP 屏蔽了站外资源的加载,导致检测脚本无法检测到站外资源的加载,这种情况下检测脚本未上报注入数据,而 CSP 上报了注入数据. 通过分析 CSP 上报的数据,发现 65%的注入数据是由被篡改的站内脚本文件加载的,35%是由被篡改的网页文件加载的. 这说明大多数注入是对脚本文件的注入,而这种注入方式往往是由带外注入主体引起的,因为采用带外注入的方式注入脚本文件是方便且节省成本的. 另外,有一些注入是第三方统计平台加载的,但无法确定是这些第三方统计平台本身加载的还是这些第三方统计平台受到注入之后加载的.

采用在服务器部署检测脚本和 HTTPS 的方案对服务器 CPU 和带宽消耗较低,但由于需要进行页面跳转,用户体验较差.

建议服务器采用部署 CSP 的方案,在测试结果中没有发现 CSP 被篡改的情况,且没有误报或漏报的情况,因此采用 CSP 方案会有一个较高的准确率. 同时采用 CSP 方案对服务器性能要求较低,成本较低,且用户体验较好.

客户端部署访问限制程序的方案的注入样本是模拟的,而访问限制程序也是基于注入样本实现的,因此这个过程中并未检测到误报和漏报. 但这并不能说明该方案的准确率,因为注入样本是会随着时间发生变化的,如果有新的注入样本出现,则该方案会产生漏报,如果旧的样本对应的域名成为一个合法网站,则该方案会产生误报. 同时,客户端加载页面时间有所提升是因为 PAC 会影响客户端性能. 如果客户端性能较差,页面加载时间则会增长. 这里是采用 PC 环境模拟手机端,而移动端处理器的性能比 PC 处理器性能差. 另外,被注入的页面加载时间却大大降低,这是因为被注入的页面会加载各种站外资源,这些资源的加载影响页面的加载时间,同时有些资源无法正常访问,这使得客户端处于一直等待响应的状态,这也影响页面加载时间. 但是使用客户端部署访问限制程序的方案后,注入资源在忽略不计的时间内被屏蔽,这使得大多数原本加载时间超过 3s 的页面在 200ms 内加载完成.

建议性能较好的客户端采用客户端部署访问限制程序的方案.

6　结　论

本文将一个面向移动端用户的访问量较大的网站作为测试平台,通过在网页部署脚本,对可能存在的 HTTP 流量注入进行检测和上报. 将这些上报的数据进行分组,整理为 124 535 个会话. 将这些会话的注入资源进行提取和分组,提取到 1 145 个资源样本和 196 个资源组. 依次对这些资源样本的内容进行分析,将注入内容分为包括广告推送在内的 7 类. 对 196 个资源组依次进行分析,最终得到 61 个注入主体. 我们发现这些注入主体和网络运营商、用户终端网络环境、地理位置等因素有关.

基于上述研究,本文提出了针对 HTTP 流量注入的屏蔽方案,包括 3 项基于服务器的方案和 1 项基于客户端的方案,并对这些方案进行安全性、性能、成本、准确率方面的测试. 测试结果表明,这 4 类屏蔽方案均能有效屏蔽 HTTP 流量注入. 其中,在服务器部署 CSP 的方案是成本低、准确率高的服务器屏蔽方案,在客户端部署访问限制程序的方案是有效的针对高性能客户端的屏蔽方案.

参考文献:

[1] 火绒安全. 百度旗下网站暗藏恶意代码劫持用户电脑疯狂"收割"流量[EB/OL]. [2017-02-28]. *http://tech.sina.com.cn/i/2017-02-28/doc-ifyavvsk3974317.shtml*.
Huorong Security. The Websites in Baidu Hide Malicious Code That Hijack Users' Computer, Aiming at Increasing Traffic [EB/OL]. [2017-02-28]. *http://tech.sina.com.cn/i/2017-02-28/doc-ifyavvsk3974317.shtml*(Ch).

[2] 毛启盈. 财联社就流量劫持事件正式起诉中国联通[EB/OL]. [2017-01-08]. *http://it.sohu.com/20170108/n478087658.shtml*.
MAO Q Y. The Financial Union Sues China Unicom for Traffic Hijacking[EB/OL]. [2017-01-08]. *http://it.sohu.com/20170108/n478087658.shtml*(Ch).

[3] CILLULA J. Five Creepy Things Your ISP Could Do if Congress Repeals the FCC's Privacy Protections [EB/OL]. [2017-03-19]. *https://www.eff.org/deeplinks/2017/03/five-creepy-things-your-isp-could-do-if-congress-repeals-fccs-privacy-protections*.

[4] BRANDOM R. Losing the ISP Privacy Fight is Only the Beginning [EB/OL]. [2017-03-29]. *http://www.theverge.com/2017/3/29/15108474/isp-privacy-rules-congress-fcc-web-history*.

[5] REIS C, GRIBBLE S D, KOHNO T, *et al*. Detecting In-Flight Page Changes with Web Tripwires [DB/OL]. [2017-05-01]. *https://www.usenix.org/legacy/events/nsdi08/tech/full_papers/reis/reis.pdf*.

[6] ZIMMERMAN P T. *Measuring Privacy, Security, and Censorship Through the Utilization of Online Advertising Exchanges* [R]. Princeton: Princeton University, 2015.

[7] WEAVER N, KREIBICH C, DAM M, *et al*. Here be web proxies[C]//*International Conference on Passive and Active Network Measurement*. New York: Springer International Publishing, 2014: 183-192.

[8] KREIBICH C, WEAVER N, NECHAEV B, *et al*. Netalyzr: Illuminating the edge network[C]//*Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. New York: ACM, 2010: 246-259. DOI:10.1145/1899141.1879173.

[9] NAKIBLY G, SCHCOLNIK J, RUBIN Y. Website-Targeted False Content Injection by Network Operators [DB/OL]. [2017-05-01]. *https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_nakibly.pdf*.

[10] Mozilla Developer Network. Network Information API [EB/OL]. [2017-04-10]. *https://developer.mozilla.org/en-US/docs/Web/API/Network_Information_API*.

[11] Malicious Web Site Labs. MWSL-hosts [EB/OL]. [2017-03-27]. *http://www.mwsl.org.cn/*.

[12] Adblock Plus. Chinese Supplement for the EasyList Filters [EB/OL]. [2017-04-10]. *https://easylist-downloads.adblockplus.org/easylistchina+easylist.txt*.

[13] Virus Total. Free Online Virus, Malware and URL Scanner[EB/OL]. [2017-04-11]. *https://virustotal.com*.

[14] Dr. Web—innovative anti-virus technologies. Comprehensive Protection from Internet Threats [EB/OL]. [2017-04-10]. *https://www.drweb.com/*.

[15] FIELDING R, RESCHKE J. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content [EB/OL]. [2014-06-01]. *https://tools.ietf.org/html/rfc7231*.

[16] Wikipedia. Content Delivery Network [EB/OL]. [2017-04-10]. *https://en.wikipedia.org/wiki/Content_delivery_network*.

[17] Firefox Telemetry. HTTPS Usage Map [EB/OL]. [2017-04-11]. *https://ipv.sx/telemetry/country-https-transactions.html*.

[18] STAMM S, STERNE B, MARKHAM G. Reining in the Web with content security policy[C]//*Proceedings of the 19th International Conference on World Wide Web*. New York: ACM, 2010: 921-930.

[19] Wikipedia. Proxy Auto-Config [EB/OL]. [2017-04-11]. *https://en.wikipedia.org/wiki/Proxy_auto-config*.

[20] Mozilla Developer Network. Performance.timing [EB/OL]. [2017-05-17]. *https://developer.mozilla.org/en-US/docs/Web/API/Performance/timing*.

[21] Google. chrome.webRequest [EB/OL]. [2017-05-17]. *https://developer.chrome.com/extensions/webRequest*.

□