软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(4):843−858 [doi: 10.3724/SP.J.1001.2013.04376]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

网页木马机理与防御技术*

张慧琳[1,2], 邹 维[3], 韩心慧[1,2]

[1](北京大学 计算机科学技术研究所,北京 100080)

[2](北京大学 互联网安全技术北京市重点实验室,北京 100080)

[3](中国科学院 信息工程研究所,北京 100093)

通讯作者: 韩心慧, E-mail: hanxinhui@pku.edu.cn

摘 要: 网页木马是一种以 JavaScript,VBScript,CSS 等页面元素作为攻击向量,利用浏览器及插件中的漏洞,在客户端隐蔽地下载并执行恶意程序的基于 Web 的客户端攻击.网页木马的表现形式是一个或一组有内嵌链接关系的页面/脚本,有漏洞的客户端在访问该(组)页面时会"过路式下载"木马等恶意程序.网页木马通过这种被动攻击模式,能隐蔽、有效地将恶意程序植入客户端,这已经成为恶意程序传播的一种重要方式.近年来,围绕网页木马的攻防博弈在持续进行.首先阐述网页木马的机理和特点,然后从检测、特征分析、防范这 3 个方面对网页木马防御方的研究进行总结和分析,最后对网页木马攻防双方的发展趋势进行讨论.

关键词: 网页木马;客户端攻击;被挂马网页;混淆;内嵌链接

中图法分类号: TP393 **文献标识码**: A

中文引用格式: 张慧琳,邹维,韩心慧.网页木马机理与防御技术.软件学报,2013,24(4):843−858. http://www.jos.org.cn/1000-9825/4376.htm

英文引用格式: Zhang HL, Zou W, Han XH. Drive-by-Download mechanisms and defenses. Ruanjian Xuebao/Journal of Software, 2013,24(4):843−858 (in Chinese). http://www.jos.org.cn/1000-9825/4376.htm

Drive-by-Download Mechanisms and Defenses

ZHANG Hui-Lin[1,2], ZOU Wei[3], HAN Xin-Hui[1,2]

[1](Institute of Computer Science and Technology, Peking University, Beijing 100080, China)

[2](Beijing Key Laboratory of Internet Security Technology, Peking University, Beijing 100080, China)

[3](Institute of Information Engineering, The Chinese Academy of Sciences, Beijing 100093, China)

Corresponding author: HAN Xin-Hui, E-mail: hanxinhui@pku.edu.cn

Abstract: Drive-by-Download is a Web-based attack that targets at downloading and executing malwares on the client side without the user's notice or consent. It usually takes HTML elements (e.g. JavaScript, VBScript, CSS) as attack vectors, and exploits vulnerabilities in browser and plugins to launch attacks. Drive-by-Download represents as an HTML page or a group of inline-linked HTML pages/scripts. After browsing these pages, vulnerable client sides will automatically download and execute malware. Through the pull-based attack mode, Drive-by-Download can effectively and secretly spread malware to clients and has become an important way to spread malware. In recent years, both the offense-side and defense-side make ongoing development. This paper first introduces the mechanisms and features of Drive-by-Download. Then the paper summarizes and discusses researches on detection, analysis and prevention of Drive-by-Download. Trends of Drive-by-Download and some possible research directions will be discussed at last.

Key words: Drive-by-Download; client-side attack; landing page; obfuscation; inline linking

* 基金项目: 国家自然科学基金(61003217, 61003216); 发改委国家信息安全专项([2010]3044); 国家 242 信息安全计划(2011A40)

收稿时间: 2012-03-23; 修改时间: 2012-07-23; 定稿时间: 2013-01-25

网页木马是近年来出现的一种新形态的恶意代码,它通常被人为置入 Web 服务器端的 HTML 页面中,目的在于向客户端传播恶意程序:客户端访问该页面时,它利用客户端浏览器及其插件的漏洞将恶意程序自动植入客户端.网页木马的表现形式是一个或一组有链接关系、含有(用 JavaScript 等脚本语言编写的)恶意代码的 HTML 页面,恶意代码在该(组)页面被客户端浏览器加载、渲染的过程中被执行并利用浏览器及插件中的漏洞隐蔽地下载、安装、执行病毒或间谍软件等恶意可执行文件.网页木马是一种客户端攻击方式,相比较蠕虫等通过网络主动进行自我复制、自我传播,网页木马部署在网站服务器端,在用户浏览页面时发起攻击.这种客户端攻击方式可以有效地绕过防火墙的检测,隐蔽地、有效地在客户端植入恶意代码,进而在用户不知情的情况下自动完成恶意可执行文件的下载、执行,国外将这种攻击方式称为 Drive-by-Download(国内直译为“过路式下载”或“偷渡式下载”).

网页木马多被用于让客户端过路式下载“盗号木马”,窃取客户端个人信息,它已经成为黑客谋求经济利益的重要工具.围绕着网页木马逐渐形成了具有一定规模、分工明确的地下经济链:股票账号、信用卡账号乃至游戏账号等都可能被盗号木马窃取,甚至网上虚拟货币也在黑客盗取的目标范围之内[1].除了形成一种相对独立的以盗号为目的的地下经济链,网页木马也作为整个互联网地下经济生态链中的一个重要环节[2,3]:作为恶意程序传播的一种重要方式,网页木马可以使客户端过路式下载任意流氓软件、僵尸程序等,攻击者从而可以在流氓软件发行商处获得推广分成或者利用大量“肉鸡”形成的僵尸网络来发动大规模攻击.

网页木马凭借其自身的独有特点成为了安全领域的学术研究热点.2005 年开始,国内外各级安全学术会议上陆续都有围绕网页木马防御的相关论文发表.围绕网页木马的攻防博弈在持续进行:攻击者不断采用一些新的手段来提高网页木马攻击成功率以及增强其隐蔽性;防御方安全研究人员随着对网页木马研究的深入,不断提出相应的检测和防范方法.

本文主要对网页木马攻防双方的思路、方法、技术进行归纳、分类及总结,旨在明确网页木马工作机制与特性,并总结防御方的研究现状以及探讨攻防双方的发展趋势,为进一步研究作参考.

本文第 1 节介绍网页木马工作机制,主要包括网页木马定义、典型攻击流程、漏洞利用机理及一些提高攻击成功率、增强自身隐蔽性方面的对抗手段.第 2 节从监测、特征分析、防范这 3 方面对网页木马防御方的研究进行总结和分析.第 3 节讨论网页木马攻防双方的发展趋势并对全文进行总结.

1 网页木马工作机制与特性

1.1 网页木马定义

网页木马是在宏病毒、木马等恶意代码基础上发展出来的一种新形态的恶意代码.类似于宏病毒通过 Word 等文档中的恶意宏命令实现攻击.网页木马一般通过 HTML 页面中的一段恶意脚本达到在客户端下载、执行恶意可执行文件的目的,而整个攻击流程是一个“特洛伊木马式”的隐蔽的、用户无察觉的过程,因此,国内研究者通常称该种攻击方式为“网页木马”.

目前,学术界对网页木马尚无一个明确的、统一的定义.Wiki 将它定义为一种用户不知情的下载[4],发生的场景可以是用户阅览邮件、访问网站页面以及点击一个欺诈性的弹出框时,等等,该定义属于字面解释,包括的内容比较宽泛,如利用社会工程学手段欺骗用户下载也属于其涵盖范围.此外,Wiki 还用 Drive-by-Install 这一术语来表示下载并安装恶意程序的过程;Google 的 Provos 等人[5]将网页木马限定为客户端在访问页面时受到攻击并导致恶意可执行文件的自动下载、执行,该定义指出了“访问页面时受到攻击”和“自动下载、自动执行恶意可执行文件”两个关键点;UCSB 的 Cova 等人[6]进一步指出,网页木马是以一段 JavaScript 代码作为攻击向量的一种客户端攻击方式,而实际上,还存在一些通过 CSS 元素、VBScript 脚本发起攻击的网页木马[7].

综上所述,本文将网页木马定义为:一种以 JavaScript,VBScript,CSS 等页面元素作为攻击向量,利用浏览器及插件中的漏洞,在客户端隐蔽地下载并执行恶意程序的基于 Web 的客户端攻击.以下是与网页木马相关的几个术语[5]:

- 攻击脚本:作为攻击向量的 JavaScript,VBScript 代码片段;

- 攻击页面(exploit page):攻击向量所在页面;
- 网页木马宿主站点(exploit site):攻击脚本/攻击页面所在站点;
- 恶意可执行文件下载站点(malware distribution site):网页木马攻击成功后,在客户端植入的恶意程序的所在站点.为了便于表述,文中将其简称为“下载站点”.

1.2 网页木马典型攻击流程

网页木马采用一种被动攻击模式(即 pull-based 模式):攻击者针对浏览器及插件的某个特定漏洞构造、部署好攻击页面之后,并不像发送垃圾邮件那样主动将内容推送给受害用户(即 push-based 模式),而是被动地等待客户端发起的页面访问请求.其典型攻击流程如图 1 所示:1. 客户端访问攻击页面;2. 服务器做出响应,将页面内容返回给客户端;3. 页面被浏览器加载、渲染,页面中包含的攻击向量在浏览器中被执行并尝试进行漏洞利用;4,5. 存在该漏洞的客户端被攻破,进而下载、安装、执行恶意程序.

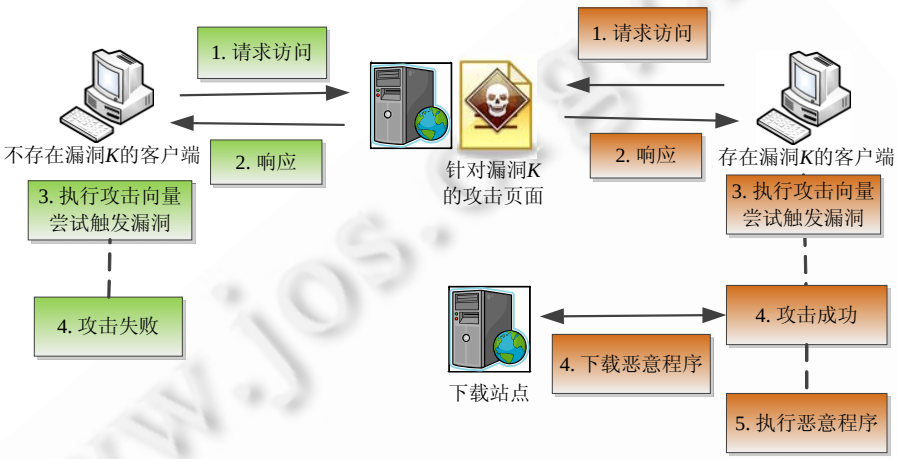

Fig.1 Attack process of Drive-by-Download

图 1 网页木马攻击流程图

从网页木马的攻击流程可以看出,网页木马是一种更加隐蔽、更加有效的向客户端传播恶意程序的手段:相比较蠕虫以主动扫描等方式来定位受害机并向其传播恶意程序,网页木马采用一种“守株待兔”的方式等待客户端主动对页面发出访问请求,这种被动式的攻击模式能够有效地对抗入侵检测、防火墙等系统的安全检查.

1.3 网页木马的漏洞利用机理

网页木马的核心在于利用客户端浏览器及其插件的漏洞获得一定权限,达到下载并执行恶意程序的目的(如图 1 所示中的步骤 3~步骤 5).其中,漏洞利用代码多用 JavaScript 等脚本语言编写,一方面在于浏览器提供了脚本语言与插件间进行交互的 API,攻击脚本通过畸形调用不安全的 API 便可触发插件中的漏洞;另一方面,攻击者也可以利用脚本的灵活特性对攻击脚本进行一定的混淆处理来对抗反病毒引擎的安全检查.网页木马利用的漏洞主要归为下述两类[6,8]:

1) 任意下载 API 类漏洞

一些浏览器插件在用来提供下载、更新等功能的 API 中未进行安全检查,网页木马可以直接利用这些 API 下载和执行任意代码,如:新浪 UC 网络电视 ActiveX 控件 DonwloadAndInstall 接口任意文件下载漏洞[9]、百度超级搜霸输入验证漏洞[10]等.

在一些复杂的攻击场景中,攻击者将多个 API 组合,完成下载和执行任意文件的目的.如 MS06-014 漏洞[11]的利用需要 3 个插件的不安全 API 分别完成恶意可执行文件的下载、本地保存、执行.

2) 内存破坏类漏洞

网页木马常常利用 JavaScript,VbScript 脚本向浏览器的内存空间填充恶意可执行指令(ShellCode),并利用该类漏洞触发执行流跳转,将执行流跳转到 ShellCode,ShellCode 负责下载和执行恶意可执行文件.

按照触发执行流跳转的原理,该类漏洞又可分为:

- use-after-free 型漏洞.空间已经释放掉的函数指针或对象指针实际上并未被删除或者置为 *null*,黑客精心构造,将指针指向的内存空间填充恶意指令或者精巧的数据,当程序的其他部分再次引用该指针时,引起恶意指令的执行或者控制逻辑的改变.典型的,如 MS09-002、“极风”、“极光”等漏洞[12];
- 溢出漏洞.一般由于插件 API 对参数长度、格式等检查不严格所致,如 API 接受了过长的参数造成缓冲区溢出,进而覆盖了函数返回地址等,在黑客的精心构造下,使得程序执行流跳转到预先放入内存的 ShellCode.典型的,如联众游戏 ActiveX 溢出漏洞、暴风影音 ActiveX 参数超长溢出、超星阅读器 ActiveX 参数溢出漏洞、real player ActiveX 参数溢出、迅雷 ActiveX 漏洞等;
- 浏览器解析漏洞.use-after-free 型漏洞和 API 溢出漏洞通常通过脚本恶意调用 API 触发,而浏览器解析漏洞一般在浏览器解析畸形构造的 HTML 文档或其中包含的 CSS 文档、XML 文档时被触发,造成执行流跳转到预先放入内存中的恶意可执行指令,如 MS08-078 Microsoft Internet Explorer 处理 XML 远程代码执行漏洞[13]等.

1.4 网页木马部署

从网页木马的典型攻击流程和漏洞利用机理可知,网页木马的核心是一个带有攻击脚本的或是一个畸形构造的攻击页面,只有当客户端访问该攻击页面时才可能被攻击.攻击者若想大规模地感染客户端主机,就需要保证攻击页面有一定的访问量:在网页木马发展初期,攻击者自己搭建站点来部署攻击页面并利用一些社会工程学方法诱使网民访问;网民的安全意识逐渐增强,使得攻击者不得不去寻找新的手段来增加攻击页面访问量,其中最主要的手段就是网页挂马.

网页挂马是通过内嵌链接将攻击脚本/攻击页面嵌入到一个正规页面或利用重定向机制将对正规页面的访问重定向至攻击页面.内嵌链接是 HTML 页面中一类特殊的超链接形式,其特点是:当浏览器访问页面时,无需用户点击,页面中的内嵌链接指向的内容就会被自动加载,如〈img〉标签等.攻击者进行网页挂马时经常利用的内嵌链接为带有 src 属性的〈iframe〉,〈frame〉和〈script〉标签:浏览器访问页面时,〈script〉标签的 src 属性指向的脚本将作为内嵌脚本被自动加载并在脚本引擎上下文中执行,〈iframe〉,〈frame〉等标签的 src 属性指向的页面也将作为内嵌页面被自动加载.攻击者常将嵌入的〈iframe〉,〈frame〉标签的 width,height 属性设置为 0 来避免被挂马的正规页面在视觉效果上发生变化.

攻击者通常利用网站服务器的漏洞获得相应权限来篡改页面、嵌入攻击脚本/攻击页面,但这种方式很难适用于那些安全防御比较严密的大型站点;由于浏览器在访问页面时会加载其整个动态视图[14](浏览器处理被访问页面时所加载的所有内嵌页面、内嵌脚本的层次关系图),攻击者可将攻击脚本/攻击页面挂载到页面动态视图中的任意位置来进行网页挂马,在第三方流量统计、广告位等处嵌入攻击脚本/攻击页面,是攻击者对一些门户网站页面进行挂马的常用手段[5].如图 2 所示,网站 *C* 的一个提供流量统计服务的页面是网站 *A* 的首页动态视图的一部分,攻击者通过〈iframe〉标签将攻击页面挂载到这个流量统计页面中(图 2 中的虚线箭头);由于内嵌链接的自动加载有递归性(即内嵌页面中的内嵌链接也会被自动加载),客户端在访问网站 *A* 的首页时会自动加载流量统计页面,随后也会加载流量统计页面中嵌入的攻击页面.

动态视图中被加入攻击脚本/攻击页面的正规页面被称为“landing page”[5](如图 2 所示中网站 *A* 的首页,实际上,在本例中,页面 *b* 和流量统计页面也是 landing page),该词形象地描绘出用户访问该页面就会加载攻击脚本/攻击页面,并开始本文第 1.2 节中描述的攻击流程.为了便于描述,本文将 landing page 称为被挂马页面,并将 landing page 所在站点称为“被挂马网站(landing site)”[5].在被挂马页面的动态视图中,通过内嵌链接/重定向连接成的、从被挂马页面到攻击脚本/攻击页面的一个自动加载链式结构被称为感染链(如图 2 中深色部分所示).

虽然网页挂马是对网站服务器中的页面进行篡改,但攻击者进行网页挂马的目的在于攻击客户端,浏览器在访问被挂马页面时,依照感染链将攻击脚本/攻击页面加载到客户端,最终让客户端自动下载、执行恶意程序.

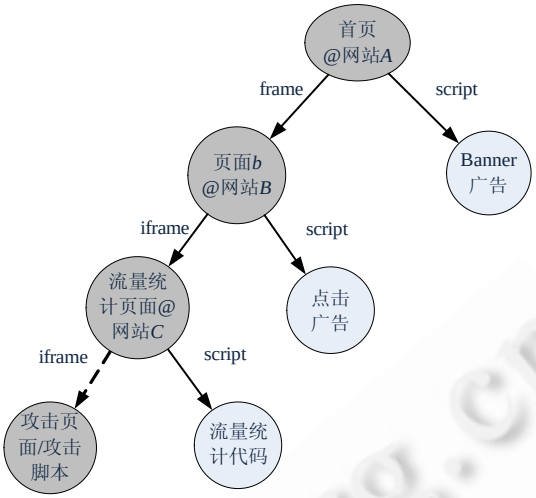

Fig.2 An example of landing page's dynamic page view and infection chain

图 2 网站 *A* 的首页被挂马后所对应的网页动态视图及感染链示例

Google 的研究指出,其搜索结果中 1.3%的页面为被挂马网页[5],网页挂马已经成为攻击者在部署网页木马时普遍采用的手段:一方面,与通过社会工程学手段诱使用户直接访问攻击页面相比,网页挂马使客户端在访问被挂马页面时按照网页木马感染链自动加载攻击脚本/攻击页面,有更好的隐蔽性;另一方面,攻击者通过一定的挂马策略可以很好地保证攻击脚本/攻击页面在客户端的加载量(如对热点事件有关页面进行挂马、对门户网站首页等访问量大的页面进行挂马等,文献[15]指出,高考前后各大高校网站广泛被挂马).

1.5 网页木马涉及的关键手段

网页木马是以页面为攻击向量载体的基于 Web 的客户端攻击方式,同时也是一种新形态的恶意代码,其组成和结构与病毒等恶意代码有很大区别.在这种新的攻击形态中,攻击者常使用一些灵活多变的技术和手段来提高网页木马攻击成功率以及躲避防御方的检测与反制.

1.5.1 提高网页木马攻击成功率的手段

网页挂马虽然使客户端访问被挂马页面时自动加载攻击脚本/攻击页面,但却无法保证攻击脚本/攻击页面被客户端加载后一定能攻击成功:一方面,攻击页面/攻击脚本针对的漏洞一般只存在于特定版本的浏览器和插件中,而客户端浏览环境各异,若攻击向量与客户端浏览环境不匹配,就会导致攻击失败(如图 1 中左侧部分所示);另一方面,即使客户端浏览环境中存在相应漏洞,但操作系统防御技术的不断升级,使得该漏洞被成功利用变得更加困难.

攻击者通过“环境探测+动态加载”模式来应对客户端浏览环境的多样性,并在攻击脚本/攻击页面中采用 Heap Spray[16](国内称为“堆喷射”)技术对抗客户端操作系统中的 ASLR(address space layout randomization,地址空间随机化)[17]防御.

- 环境探测+动态加载

为了应对客户端浏览环境的多样性,攻击者采用一种 all-in-one 的方式将针对不同漏洞的攻击代码全部包含进单个攻击页面中.但这种方式导致大量攻击代码在浏览器中执行,容易使浏览器反应迟缓,引起用户警觉.为了在提高网页木马的攻击效率和成功率的同时保证攻击的隐蔽性,攻击者采用了“一个探测页面+多个攻击脚本/攻击页面”的“环境探测+动态加载”模式[18]:其中,一个攻击脚本/攻击页面仅攻击单个漏洞,攻击者拥有针对不同漏洞的攻击脚本/攻击页面;探测页面中的脚本利用浏览器提供的 JavaScript API 对客户端浏览器版本、插件版本等进行探测,并使用 DOM[19] API(如 document.write 等)动态加载与探测结果相对应的攻击脚本/攻击页面.

“环境探测+动态加载”型网页木马中充分利用了 JavaScript 等客户端脚本的灵活性,能够根据客户端的不同配置动态地、有选择地加载不同的攻击脚本/攻击页面.这种方式提高了网页木马对不同客户端浏览环境的攻击成功率,同时也保证了攻击的效率和隐蔽性.

- Heap Spray

Heap Spray 主要用于辅助内存破坏类漏洞的利用.ASLR 等防御技术使得该类漏洞被触发时无法准确地将执行流跳转到 ShellCode 所在位置,从而导致攻击失败.而 Heap Spray 的原理为:在 ShellCode 前添加大量无用指令(如 NOP 指令)构成的着陆区(sledge),并通过 JavaScript 字符串赋值将这种“着陆区+ShellCode”结构大量填充进堆空间;网页木马通过漏洞利用将执行流跳至堆空间后,即使没有准确跳转至 ShellCode 的位置,由于此时堆空间中的大部分区域都已被着陆区覆盖,执行流有相当大的可能会落在着陆区,执行流执行着陆区中一系列的 NOP 无用指令之后,进而会执行 ShellCode.

Heap Spray 能够有效绕过 ASLR,提高了网页木马的攻击成功率.

1.5.2 增强网页木马自身隐蔽性的手段

网页木马是一个或一组有链接关系、含有(用 JavaScript 等脚本语言编写的)恶意代码的 HTML 页面,页面间通过复杂的结构与多种灵活机制相连接,更容易隐蔽自身以绕过检测.攻击者常通过混淆免杀、人机识别、动态域名解析等技术手段来躲避检测与追踪.

- 混淆免杀

攻击者常对攻击脚本作各种混淆、加密,消除其原有的特征,以躲避特征扫描工具的检测.具体的混淆方式有,将字符串中填充大量垃圾字符,以及采用十六进制编码、Unicode 编码、escape 函数编码和一些字符串处理函数,甚至可以对一段脚本进行多次混淆.网页木马利用脚本的动态特性,在解释执行混淆脚本的过程中逐步还原出攻击脚本的真实形态,如用正则表达式来代替或去掉垃圾字符、用 eval 函数执行解码后的字符串.攻击者常使用混淆免杀机制来对抗一些基于静态特征的检测.

- 人机识别

为了躲避防御方的自动化检测,网页木马还常常采用一些人机识别手段,认定客户端是人工浏览行为后再触发进一步的感染(如用户点击某按钮之后才触发攻击);通过设置、检查植入在客户端的一个标识参数等防重入机制避免网页木马宿主站点中多个攻击页面被同一主机重复访问,进而避开防御方对该站点的自动化检测.

- 动态域名解析

动态域名解析是目前比较常见的 DNS 解析技术,即将一个域名解析到一个 IP 动态变化的主机.攻击者滥用动态域名解析服务,申请大量免费动态域名,根据动态域名解析机制随意改变网页木马宿主站点的位置而不影响用户通过域名对攻击页面的访问.这种流窜作案方式增加了防御方的追踪难度.

网页木马为了躲避检测及增加隐蔽性,还采用了多种其他机制,如:使用随机的 URL 参数来躲避黑名单过滤;将 JavaScript 和 VBScript 混用来躲避基于单一脚本分析的检测.相比较传统的病毒,网页木马具有独特的、复杂的结构和多种灵活机制,更容易绕过检测与追踪,能够更隐蔽地进行大规模的感染,因此具有更大的危害性.

1.6 网页木马特性小结

网页木马是一种基于 Web 的客户端攻击方式,也是在传统木马、病毒等基础上发展而来的一种新形态的恶意代码,但与传统木马、病毒相比,其在结构与形态等方面有一些自己独有的特点:

(1) 基于 Web 的被动式攻击模式:攻击者通过网页挂马将网页木马广泛置入 Web 服务器端的 HTML 页面中,并采用“守株待兔”的被动方式等待客户端在浏览被挂马页面时加载网页木马.随着近年来全球网页访问量呈爆炸式增长,这种基于 Web 的被动式攻击模式使网页木马能够十分隐蔽并有效地感染大量客户端;

(2) 两级感染模式:网页木马本身是一种新形态的恶意代码,目的在于使客户端自动下载、执行恶意程序.在网页木马典型的攻击流程中涉及两种不同类型的恶意代码形态——作为攻击向量的 JavaScript,

VBScript,CSS 等页面元素以及最终被下载、执行的恶意程序,这两种形态分别对应于客户端感染的两个阶段:

i. 漏洞利用阶段:网页木马被客户端加载后,攻击向量利用内存破坏类漏洞将执行流跳转到 ShellCode 或者直接利用任意下载 API,在客户端下载、执行盗号木马或僵尸程序等恶意程序;

ii. 恶意程序执行阶段:下载的盗号木马或僵尸程序等恶意程序,窃取客户端的帐号等隐私信息或使客户端成为“肉鸡”加入僵尸网络;

(3) 通过内嵌链接构成的树状多页面结构:攻击者通过网页挂马将网页木马挂载到被挂马网页的动态视图中,客户端在访问被挂马网页时会沿着网页木马感染链自动加载网页木马;“环境探测+动态加载”机制使网页木马可以在探测客户端操作系统、浏览器及插件版本等信息后,有针对性地动态加载漏洞利用代码来提高攻击效率及隐蔽性.被挂马页面、探测页面和多个攻击脚本/攻击页面之间最终形成了一种灵活多变的、通过内嵌链接构成的树状多页面结构;

(4) 灵活多变的隐蔽手段:网页木马以 JavaScript 等页面元素作为攻击向量,以 HTML 页面作为攻击向量载体发起对客户端的攻击;攻击者通过灵活多变的手段来隐蔽自身,如对脚本进行混淆以掩盖攻击意图并躲避检测,并采用人机识别、动态域名解析等手段来躲避防御方的检测与反制.

2 网页木马防御技术研究

攻击者通过网页挂马将网页木马挂载到可信度较高的正规页面中,客户端在访问这些被挂马页面时,攻击页面/攻击脚本作为被挂马页面动态视图的一部分被自动加载并利用浏览器的漏洞在客户端下载、执行僵尸程序等各类恶意可执行代码.整个感染过程由客户端访问被挂马页面时开始,并隐蔽在用户正常的页面浏览活动中,这种感染方式与蠕虫等相比,能够更加隐蔽、有效地将恶意程序植入客户端.

网页木马是一种高效、隐蔽的客户端攻击方式,攻击者对互联网上大量页面进行网页挂马,破坏了互联网的正常浏览环境.为了有效应对该种安全威胁,防御方重点关注如下几个方面:

1) 在互联网范围内大规模进行网页挂马检测,掌握攻击者进行网页挂马的范围及趋势,使网站管理员及时做出应急响应,移除页面中的恶意内容.这一过程涉及到一些具体的网页木马检测技术;

2) 通过对网页木马样本的分析,了解网页木马新的对抗手段,提取网页木马新的特征,指导防御方更好地进行检测和防范;

3) 网页木马防范.网页木马是一种部署在网站服务器中、针对客户端实施的攻击;如何在网页木马部署、实施攻击的各个阶段对其进行有效防范,是防御方的研究重点.

本节就近年来防御方在这 3 方面的研究工作进行阐述与讨论.

2.1 大规模网页挂马检测思路及检测方法

在互联网中大规模进行页面检查,检测出被挂马页面,是网页木马防御的重要环节:一方面,可以通知被挂马网站的管理员及时清除页面中嵌入的恶意内容,从而改善互联网浏览环境;另一方面,有助于让防御方掌握网页挂马的范围、趋势以及捕获最新的网页木马样本.

2.1.1 大规模网页挂马检测思路

大规模网页挂马检测的基本思路为:使用爬虫爬取互联网页面,将爬取到的 URL 输入到检测环境——客户端蜜罐中进行网页木马检测.客户端蜜罐(client honeypot)这一概念首先由国际蜜网项目组(The Honeynet Project)的 Spitzner 针对网页木马这种被动式的客户端攻击而提出.在网页挂马检测时,客户端蜜罐根据 URL 主动地向网站服务器发送页面访问请求,并通过一定的检测方法分析服务器返回的页面是否带有恶意内容.

在互联网中大规模进行网页挂马检测有两个关键点:

1) 提高爬虫爬取的覆盖率,这方面可以通过采用大规模的计算平台、设计均衡的并行爬取分配策略等来实现,本文不作过多阐述;

2) 在客户端蜜罐中实现高效、准确的网页木马检测方法,本文将在第 2.1.2 节具体介绍各种检测方法并

加以讨论.

2.1.2 网页挂马检测方法

客户端蜜罐可分为高交互式和低交互式两种:高交互式客户端蜜罐中采用一个真正的带有漏洞的浏览器与网站服务器交互,并采用基于行为特征判定等方法进行网页木马检测;低交互式客户端蜜罐则轻量级地模拟出一个浏览器,其中,页面获取模块模拟浏览器向服务器发出页面访问请求,页面检测模块采用基于反病毒引擎扫描、基于统计或机器学习、基于漏洞模拟等方法对获取的页面进行检测.

- 基于反病毒引擎扫描的检测

基于反病毒引擎扫描是研究人员进行网页挂马检测时较早使用的方法[20,21],该方法主要基于规则或特征码匹配来检测页面中是否含有恶意内容.

该方法的优点在于可以快速地对页面进行检测,但其缺点在于存在较高的漏报率:一方面,仅仅依赖一种纯静态的特征匹配无法对抗网页木马为了提高隐蔽性而普遍采用的混淆免杀机制,根据互联网安全技术北京市重点实验室在2008年底作的统计发现[1],9款国内外主流反病毒软件对在中国互联网上发现的网页木马样本最高仅达到 36.7%的检测率;另一方面,研究人员通常仅对单页面进行扫描[20,21],而不进一步检测页面动态视图中的内嵌脚本/内嵌页面,从而无法有效检测出被挂马页面.

- 基于行为特征的检测

基于行为特征的网页木马检测方法通常被用于高交互式客户端蜜罐中:即在检测环境中安装真实浏览器和一些带有漏洞的插件,驱动浏览器访问待检测页面,通过监测页面访问时注册表变化、文件系统变化、新建进程等行为特征来判定页面是否被挂马.为了便于感染后快速恢复以及防止恶意程序在本地网络中的传播,高交互式客户端蜜罐一般部署在虚拟机中并作一定的网络隔离.

2006年,微软研究院的 HoneyMonkey 项目[22]以及华盛顿大学 Moshchuk 等人[23]的 SpyCrawler 采用该方法均发现了一定数量的被挂马网页,从而验证了网页挂马在互联网上的现实存在.此外,还有一些开源的高交互式客户端蜜罐,如 HoneyClient[24]以及国际蜜网项目组新西兰小组的 Capture-HPC[25].

基于行为特征的网页木马检测方法采用了一个真实的浏览器,页面中的任何内嵌元素以及递归的内嵌元素都会在访问待检测页面的过程中被顺次加载,混淆脚本也在“解释执行”过程中被自动解混淆,该方法有效地弥补了基于反病毒引擎扫描的两点不足;而且,该方法根据攻击结果检测网页木马,所以误报率比较低.但是,该方法也存在一定的局限性:高交互式客户端蜜罐中安装的浏览器和插件版本与攻击向量不匹配会导致网页木马攻击失败,造成检测系统无法监测到行为特征,形成一定的漏报.此外,该方法采用一个真实的浏览器并需要监测系统行为,往往有较高的系统代价;并且,该方法也需要较高的时间代价等待攻击成功并出现行为特征(每个页面大约需要2分钟的检测时间[22]).最后,为了对抗基于行为特征的检测,攻击者在网页木马中加入了一系列的对抗手段,如需要一定的用户交互才触发攻击、使用定时炸弹(time bomb)来延迟攻击、使用按概率方式随机触发攻击的策略等,这些都给基于行为特征的网页木马检测带来了一定的困难.

- 基于统计或机器学习的检测

由于网页木马经常对恶意脚本进行混淆来躲避基于特征码的检测,一种检测思路是按照页面的混淆程度来进行恶意性判断.文献[26,27]提出了基于判断矩阵法的恶意脚本检测方法,但该方法仅对经过 encode 和 escape 函数混淆的恶意脚本有效.随着越来越多的大型网站为保障代码知识产权都对自己的脚本进行了一定的混淆或加密,因此,这类按照混淆程度进行恶意性判定的方法会带来较多的误报.

Seifert 等人[28]提取页面中更多的静态特征进行机器学习来对页面进行分类,提取的特征主要有:〈applet〉以及〈object〉标签的数量、〈script〉标签数量、带有 src 属性的〈iframe〉〈frame〉标签的数量以及 3XX 跳转、meta-refresh 刷新、〈iframe〉的宽度高度以及脚本中使用的字符串编码/解码函数的数量等.实验结果显示,采用这种方式有5.88%的误报率但却有46.15%的漏报率.UCSB 的 Kruegel 等人[29]的工作与 Seifert 相似,从他们工作的结果来看,基于静态特征进行机器学习的检测方法有着较高的漏报率.

Kruegel 等人在文献[6]中提取脚本执行时的动态特征进行机器学习,其在一个浏览器模拟框架 HTMLUnit

中访问待检测 URL 并监测脚本执行,提取页面跳转、脚本混淆、字符串操作、插件函数执行这 4 方面的 10 个动态特征,并基于这 10 个动态特征进行机器学习.实验结果表明,其漏报率为 0.2%.与仅提取页面静态特征相比,基于脚本动态特征的机器学习极大地降低了漏报率.

- 基于漏洞模拟的检测

该类方法的典型代表是 PHoneyC[7].PHoneyC 在低交互式客户端蜜罐环境中解析页面并用一个独立的脚本引擎执行提取出的脚本,通过在脚本引擎上下文中模拟出一些已知的漏洞插件,并结合运行时参数检查来检测恶意脚本.

作为低交互式客户端蜜罐,PhoneyC 比高交互式客户端蜜罐更迅速、更容易加载(模拟)更多的漏洞模块甚至同一漏洞模块的不同版本.但是 PhoneyC 也具有其自身局限性:由于采用一个独立的脚本引擎,脚本执行过程中常常由于缺少页面上下文环境或浏览器提供给脚本的一些 API 而导致脚本执行失败、检测被迫停止.此外,PhoneyC 只能模拟已知的漏洞插件,无法检测利用零日漏洞的恶意脚本.

在网页挂马检测中,低交互式客户端蜜罐中的基于反病毒引擎扫描、基于统计或机器学习、基于漏洞模拟等方法和高交互式客户端蜜罐中的基于行为特征的检测方法各有优缺点:低交互式客户端蜜罐的实现和配置灵活、便于扩展;高交互式客户端蜜罐主要根据行为特征进行检测,误报率相对较低,但时间代价和系统代价比较大.

为了满足大规模的网页挂马检测的高效、准确这两方面的要求,研究人员往往将多种检测方法相结合:Google 的 Provos 等人[5]在预处理模块中提取页面静态特征进行机器学习以快速过滤出可疑页面,之后用高交互式检测模块进行基于行为特征的确认;UCSB 的 Wepawet 系统[6,29]首先采用基于页面/脚本静态特征的机器学习方法快速过滤出可疑页面,之后用基于脚本动态特征的机器学习方法进行恶意性检测.

2.2 基于网页木马特征分析的反制与追踪

网页木马结构复杂、对抗机制灵活多变,防御方有必要对捕获到的网页木马样本进行特征分析:一方面可以把握网页木马的变化发展趋势,另一方面也可以了解网页木马最新的对抗机制.分析到的网页木马特征可被用来指导网页木马的检测和防范.

网页木马以页面为载体部署在网站服务器中,内容和结构随时间推移呈现高度变化[15].为了保证分析的一致性,需要对检测到的网页木马场景进行保存,并在研究人员进行特征分析时通过一种“重放”的机制还原攻击场景.Chen 等人[18]将网页木马场景定义为一个四元组(μ,V,E,T),其中,μ代表检测到的某个被挂马页面;V 代表客户端在访问被挂马页面时加载的内嵌页面、脚本等所有资源;E 代表μ,V 所有元素之间的内嵌链接/重定向关系;T 表示网页木马攻击成功后,最终在客户端下载的恶意可执行文件.文献[18]中提出了一种基于网站服务器代理的网页木马场景收集-重放方法:网页木马场景收集模块用一个高交互式的客户端蜜罐来尽量触发网页木马场景,并将场景中的各个页面、脚本按照各自的 URL 命名保存在一个用 polipo 实现的代理服务器的文件系统中.研究人员在进行样本分析时,需要将代理设定为该代理服务器,该代理服务器的重放模块处理 HTTP 请求,将保存的资源返回给分析者.

在网页木马场景收集与重放的基础上,研究人员主要采用人工的方法分析网页木马的漏洞利用过程,并用一些自动化的分析方法进行网页木马感染链的识别与构建,以及网页木马家族的识别与分析.

- 网页木马感染链的识别与构建

Provos 等人[5]在其高交互式客户端蜜罐检测网页挂马的基础上,监听浏览器访问被挂马页面时所有 HTTP 请求报文中的 Referer 字段以构建页面间的内嵌链接关系.由于 Referer 字段不是 HTTP 请求报文中的必须字段,Provos 等人还提出一种补充方法:解析出页面中的〈iframe〉,〈script〉等标签,与浏览器实际加载的 script 或 iframe 的 URL 地址进行匹配,以得到页面与页面、页面与脚本之间的内嵌关系.最后定位出攻击页面,并采用回溯的方式构建出被挂马网页的感染链.

Wang 等人[22]在其高交互式网页挂马检测环境中使用 BHO(browser helper object,浏览器辅助对象)捕获浏览器访问被挂马页面时各个加载页面之间的链接关系,并同样用回溯的方式构建被挂马网页的感染链.在研究

过程中,Wang 等人发现攻击者对攻击页面的命名有一定的规则性,如以漏洞的名字命名.

- 网页木马家族的识别与分析

Provos 等人在 10 个月内大规模进行网页挂马检测[5],检测数据表明:300 万个被挂马页面对应 9 000 个恶意可执行文件下载站点.由上述数据可知,在整个互联网中,被挂马页面和下载站点是一个大比例的多对一关系,因此,各个被挂马页面的感染链应该呈现出一种家族式的聚类关系.

Lee 等人[30]对收集到的被挂马页面的感染链进行聚类,并在聚类后定位各个家族中网页木马的核心服务器,从该服务器上的一些已知链接地址提取 URL 正则表达式特征,并用该特征指导网页木马检测.

韩等人[31]采用一种基于频繁子树的挖掘算法挖掘网页木马场景中的结构特征,并用该结构特征辅助基于行为特征的网页木马检测方法,弥补了高交互式客户端蜜罐由于检测环境中插件缺失等原因造成的漏报,并提高了总体的检测效率.

2.3 网页木马防范技术

网页木马的部署、攻击实施涉及多个环节,防御方提出了各种不同思路的网页木马防范方法与技术,本文根据网页木马防范位置的不同,将它们分为网站服务器端的网页挂马防范、基于代理的网页木马防范、客户端的网页木马防范这 3 类.

2.3.1 网站服务器端网页挂马防范

为了保证攻击脚本/攻击页面的客户端加载量以及增强隐蔽性,攻击者对互联网上大量页面进行网页挂马,因此,网站服务器端的挂马防范就成为了网页木马防范中的第一个环节.网页挂马有多种途径[32,33],主要包括利用网站服务器系统漏洞、利用内容注入等应用程序漏洞、通过广告位和流量统计等第三方内容挂马等.

利用网站服务器端系统漏洞来篡改网页内容,是常见的一种网页挂马途径:攻击者发现网站服务器上的系统漏洞,并且利用该漏洞获得了相应权限后,可以轻而易举地篡改页面.网站服务器端可以通过及时打系统补丁以及部署一些入侵检测系统来增强自身的安全性.

攻击者还常利用应用程序中的 XSS(cross-site-script,跨站脚本[34])、SQL 注入等漏洞,将恶意内嵌链接嵌入到页面中.研究者除了提出一些针对 XSS,SQL 注入的漏洞挖掘方法[35,36],还提出了一些防范方法[37,38].如,Mike 等人[37]认为,XSS 等注入攻击是由于网站服务器与浏览器对用户提交的内容理解不一致所造成的,他们提出了一种使浏览器直接按照网站服务器对用户输入的理解进行页面渲染的方法.虽然目前学术界对 XSS,SQL 注入的检测、防范研究比较多,但却仍然没有一种 one-fit-all 的方案.

攻击者挂马的途径多种多样,除了利用网站服务器本身的系统漏洞及应用服务中的注入漏洞,攻击者还可通过网页中的广告位及流量统计等第三方内容进行网页挂马.网站服务器端在网页挂马防范中,除了应关注系统及应用上的安全漏洞,也有必要对页面中的第三方内容进行一定的安全审计.

2.3.2 基于代理的网页木马防范

基于代理的网页木马防范是在页面被客户端浏览器加载之前,在一个 shadow 环境(即代理)中对页面进行一定的检测或处理.

- “检测-阻断”式的网页木马防范方法

“检测-阻断”式的网页木马防范方法基于第 2.1.2 中的方法在代理处进行网页木马检测,若发现页面存在网页木马,则阻断客户端对该页面的加载.

华盛顿大学的 Alex 等人基于之前的 SpyCrawler[23],在 SpyProxy[39]中使用高交互式客户端蜜罐部署一个 shadow 环境,客户端访问的任何页面都首先在该代理处用基于行为特征的检测方法进行网页木马检测,若判定访问的页面被挂马,就给客户端返回一个告警.“检测-阻断”式的网页木马防范方法要做到检测的有效性、用户体验的透明性,即,既要在代理处有效检测出被挂马页面并阻止客户端浏览器加载该页面,同时也不能使客户端在用户体验上有明显差别.SpyProxy 均不能满足这两点:一方面,高交互式客户端蜜罐本身就存在高系统代价、高时间代价;另一方面,高交互式客户端蜜罐浏览环境的单一性导致检测不完备,并且无法检测到需要用户交互(如鼠标点击)等操作才能触发的攻击,这些因素均会导致漏报,不能使客户端完全免于网页木马攻击.

基于机器学习的网页木马检测及防范也是一种“检测-阻断”式的网页木马防范方法.Rieck 等人[40]在代理处提取 JavaScript 脚本的动态、静态特征进行机器学习.具体来说,其提取脚本的静态词法结构和细粒度的动态行为(如字符串赋值等操作),并基于 *n*-gram 将词法结构和动态行为转换为特征向量,之后,基于支撑向量机对页面进行分类.基于机器学习的网页木马检测及防范方法虽然能在一定程度上检测出网页木马,但是也不可避免地存在漏报,客户端仍然存在被攻击的可能.

- 基于脚本重写的网页木马防范方法

Charles 等人提出的 BrowserShield[41]在代理处并不判定页面是否含有恶意内容,而是重写页面中的脚本,用一个自定义的脚本库对页面脚本中函数调用、属性获取等操作进行封装,封装函数中包含一些实时的安全检查代码,主要基于已知漏洞函数的参数超长等特征进行网页木马检测.被重写后的页面在被客户端加载时会自动执行封装函数中定义的安全检查,若发现与已知漏洞特征相匹配,则终止该段脚本执行.与在代理处进行网页木马检测的网页木马防范方法相比,BrowserShield 在代理处只进行页面重写,根据已知漏洞特征插入实时检查点.这类基于已知漏洞特征的网页木马防范方法的明显缺点是无法防范利用未知漏洞的网页木马.

- 基于浏览器不安全功能隔离的网页木马防范方法

Chen 等人在 WebShield[42]中提出了一个利用代理进行浏览器不安全功能隔离的网页木马防范框架.他们认为,页面解析器、脚本引擎等属于浏览器中的不安全模块,提出了网页木马防范的一条新思路:将这些不安全模块隔离在代理处,客户端浏览器只负责将渲染效果呈现.具体实现可如下描述:在代理处完成页面解析、脚本执行等操作,用 JavaScript 描述当前页面的 DOM 树结构,并将该段 JavaScript 代码传递给客户端浏览器;客户端浏览器通过执行该段脚本重构出 DOM 树结构,最终展现出与直接访问页面一样的效果.Chen 等人还重写页面中(鼠标点击等)事件响应机制,来支持客户端浏览器和代理端浏览器进行事件响应通信.

该种方法试图由代理负责解析页面和执行脚本,但却带来了较大的时间代价,不适于实际应用.

2.3.3 客户端网页木马防范

研究人员提出了一些直接在客户端进行网页木马防范的方法,如 URL 黑名单过滤、浏览器安全加固、操作系统安全扩展等.

- URL 黑名单过滤

Google 将基于页面静态特征进行机器学习的检测方法与基于行为特征的检测方法相结合,对其索引库中的页面进行检测[5],生成一个被挂马网页的 URL 黑名单,Google 搜索引擎会对包含在 URL 黑名单中的搜索结果做标识.

基于 URL 黑名单过滤的最大问题在于时间上的不实时以及范围上的不全面:文献[43]显示,被挂马页面的数目每月都会有一定的增加,虽然 Google 周期性地检测页面,但一个页面很可能在被 Google 判定为良性之后被挂马,用户随后浏览该页面时就可能遭到攻击.尽管 Google 爬取了大量页面并对其进行检测,但仍无法保证 100%的覆盖面[15].为了实现有效的客户端防护,必须采用一种实时的防范机制.

- 浏览器安全加固

研究者主要通过在浏览器中增加 HeapSpray/Shellcode 检测和已知漏洞利用特征检测等功能来实现浏览器的安全加固.

Egele 等人[44]修改了 Mozilla Firefox 浏览器的脚本解析引擎 SpiderMonkey[45],在其中增加了对 ShellCode 的检测.具体实现为:在脚本引擎中监测每次字符串赋值,采用 libemu[46](libemu 是一个开源的 ShellCode 检测和分析工具,它使用“启发式 GetPC 探测”的策略来判断目标内存区域是否含有 ShellCode)检测内存区域是否包含 ShellCode.一旦 libemu 检测到浏览器内存空间存在 ShellCode,脚本引擎就停止脚本的后续执行,防止客户端受到攻击.

微软的 Livshits 等人[47]侧重于对 HeapSpray 着陆区的检测,他们在 Mozilla Firefox 浏览器的内存管理模块中的 malloc,calloc,realloc,free 等函数处插入了安全检查,用一个轻量级的模拟器扫描堆空间识别出合法的 X86 指令并对其反汇编,构建出控制流图.基于该控制流图以及“攻击者想从代码块的任意位置跳转到堆空间并执行

ShellCode”这一 Heap Spray 特征检测着陆区的存在.Ding 等人[48]提出了文献[47]存在的问题,并给出了解决方案.Livshits 等人基于文献[47]又提出一种静态检测 HeapSpray 的方案[49]:将脚本拆解成抽象语法树结构,并用机器学习分类.基于 HeapSpray/Shellcode 检测的客户端防范只针对利用内存破坏类漏洞的网页木马,因此并不是一种 one-fit all 的方法.

Song 等人[50]用有限状态自动机分别描述已知漏洞的利用特征,在 IE 浏览器中监测组件间的通信流,判定模块间交互(特别是源自脚本引擎的交互)是否匹配某个自动机模型.该种方法不能检测并防范利用零日漏洞的网页木马.

- 操作系统安全扩展

Lee 等人[51]认为,浏览器由于存在各种漏洞,是一个不安全的环境,但客户端的操作系统是一个相对安全的环境.他们提出的 BLADE 对操作系统作一定的安全扩展,阻断网页木马攻击流程中未经用户授权的恶意可执行文件下载、安装/执行环节:任何通过浏览器进程下载的可执行文件都被放入一个虚拟的、权限受限的隔离存储空间,只有经过用户确认的下载文件才会被转移到真实的文件系统中.BLADE 虽然在一定程度上阻断了恶意可执行文件在客户端的下载、执行,但是并未阻止网页木马被客户端加载和执行.

3 讨论与总结

围绕网页木马的攻防博弈一直在持续,本文第 2 节所描述的防御方各种检测和防范方法各自都有一些固有的缺陷.研究人员和浏览器厂商相继提出了一些新的浏览器安全架构和安全方案来对抗网页木马,但是网页木马依然作为一种现实的安全威胁存在,并有新的发展趋势.

3.1 围绕浏览器安全架构展开新一轮博弈

一些研究者认为,网页木马能够攻击成功的一部分原因在于现有的浏览器架构没有一定的隔离机制,将同一页面中不同源的内容一起放入同一进程或保护域.为此,他们提出了新的浏览器架构[52−54],试图对页面中不同源的内容实行安全隔离.然而,这些与实际的应用还有一定的距离.

各大浏览器厂商也分别提出自己的安全方案[55,56].Chrome 是第一个引进多进程架构的浏览器,紧接着,微软在 IE8 中以及 Mozilla 在 FireFox 中都实现了多进程架构.虽然各个厂商的进程隔离实现方案有所不同,但基本的思想是将各个 Tab、窗口、插件的运行与浏览器主体进程分开.虽然多进程架构浏览器以及微软在 Vista 及之后的操作系统中加入的 DEP,UAC 等安全策略使客户端浏览环境的安全性得到了一定程度的提升,但是攻击者仍然能够绕过这些安全机制实施网页木马攻击[57].围绕浏览器安全架构,网页木马攻防双方又开始了新一轮的博弈.

3.2 网页木马的发展趋势

- 网页木马的利用向“大众化”发展

网页木马的利用,近年来有着从“专业化”向“大众化”的发展趋势:早期,攻击页面的编写及网页挂马需要具备一定攻防技术基础的黑客才能完成;而现在,普通网民也可以随意构建并发布网页木马.这种变化趋势在于两方面原因:一是大量的网页木马自动构建工具的存在,如在著名黑客会议 Black Hat 和 DEF CON 上发布的 drivesploit[58]等,普通网民仅简单操作这些工具便可定制出针对特定漏洞的网页木马.另一方面,普通网民不需要掌握任何复杂的网页挂马手段,仅仅通过社交网站就可以大范围地推送恶意链接[59];加之 Twitter、新浪微博等会对用户上传的 URL 做短链接处理,这种恶意链接有很强的隐蔽性.

随着网页木马的利用向“大众化”的发展,网页木马在互联网上的分布更加广泛.一个可能的研究方向是根据网页木马自动生成工具的指纹特征进行网页木马检测与防范.

- 攻击向量载体向 PDF,Flash 文档格式扩展

网页木马以 HTML 页面作为攻击向量载体,在浏览器加载、渲染 HTML 页面时,利用浏览器及其插件的漏洞实现恶意程序的下载、执行.近年来,攻击者开始在 PDF 和 Flash 等文档中嵌入恶意脚本,利用 PDF,Flash 阅读

器的漏洞来下载、执行恶意程序.攻击向量的载体已经从 HTML 页面扩展到 PDF,Flash 等文档.通过 PDF 文档进行客户端攻击,呈一种上升趋势[60].

以 PDF,Flash 为攻击向量载体进行的客户端攻击已经开始得到学术界的关注.虽然 PDF 和 Flash 在文档格式上与 HTML 页面有所区别,但攻击手段本质上都是在文档中嵌入恶意脚本等攻击向量,利用浏览器/阅读器中的漏洞实现恶意程序的自动下载和执行.对于该类攻击的防御,可以借鉴网页木马防御中的一些思路,如 Tzermias 等人[61]借鉴 PhoneyC 的思路检测恶意 PDF 文档,文献[62]检测恶意 Flash 文件.具有创新性的研究在于提出一种通用的能够应对各种类型恶意文档的防御方法.

- 目标攻击平台向手机平台扩展

近年来,智能手机的市场份额在逐步扩大,技术也在不断发展.智能手机浏览环境的逐步发展给用户带来了越来越好的使用体验,但同时也将攻击者的攻击目标吸引到了手机平台.据统计,iPhone,Android 等主流手机平台均存在大量的安全漏洞,而浏览环境的安全漏洞又占据了其中的大部分.2007 年 iPhone 首次发布后不久,便被攻击者通过 Safari 浏览器上的漏洞攻破,而在 2010 年 3 月 Pwn2Own 全球攻击者大赛上,两名欧洲黑客仅用 20s 的时间便通过 Safari 浏览器成功攻破 iPhone[63].只要用户用智能手机中特定版本的浏览器访问攻击者精心构造的页面,就可能触发恶意代码在智能手机平台上自动执行[64].目前,针对手机平台的网页木马虽然没有大规模爆发,但由于手机平台浏览环境中存在大量漏洞,针对手机平台的网页木马仍然是一种潜在的安全威胁,值得研究人员关注.

3.3 总 结

网页木马作为恶意程序传播的一种重要方式,能够在客户端访问页面的过程中高效、隐蔽地将恶意程序植入客户端,基于 Web 的被动式攻击模式使网页木马能十分隐蔽并有效地感染大量客户端,通过内嵌链接构成的树状多页面结构以及灵活多变的隐蔽手段,使网页木马在结构和组成等方面与一些传统的恶意代码形态有所区别.安全研究人员基于对网页木马机理、特点等的把握,在挂马检测、特征分析、防范等方面提出了应对方法.围绕网页木马的攻防博弈仍在继续,网页木马在载体和攻击平台上的扩展也给研究人员带来了新的挑战与机遇.对于安全研究人员来说,与网页木马的对抗是一项任重而道远的工作.

References:

[1] Zhuge JW, Holz T, Song CY, Guo JP, Han XH, Zou W. Studying malicious websites and the underground economy on the Chinese Web. In: Johnson ME, ed. Proc. of the Managing Information Risk and the Economics of Security. Berlin, Heidelberg: Springer-Verlag, 2009. 225−244. [doi: 10.1007/978-0-387-09762-6_11]

[2] Caballero J, Grier C, Kreibich C, Paxson V. Measuring pay-per-install: The commoditization of malware distribution. In: Proc. of the 20th USENIX Security Symp. Berkeley: USENIX Association, 2011. http://dl.acm.org/citation.cfm?id=2028067.2028080

[3] Polychronakis M, Mavrommatis P, Provos N. Ghost turns zombie: Exploring the life cycle of Web-based malware. In: Proc. of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET). Berkeley: USENIX Association, 2008. http://dl.acm.org/citation.cfm?id=1387709.1387720

[4] Wikipedia. Drive-By download. 2005. http://en.wikipedia.org/wiki/Drive-by_download

[5] Provos N, Mavrommatis P, Rajab MA, Monrose F. All your iFRAMEs point to us. In: Proc. of the 17th USENIX Security Symp. Berkele: USENIX Association, 2008. 1−15. http://dl.acm.org/citation.cfm?id=1496711.1496712

[6] Cova M, Kruegel C, Vigna G. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In: Proc. of the 19th Int'l Conf. on World Wide Web (WWW). New York: ACM Press, 2010. 281−290. [doi: 10.1145/1772690.1772720]

[7] Nazario J. PhoneyC: A virtual client honeypot. In: Proc. of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET). Berkeley: USENIX Association, 2009. http://dl.acm.org/citation.cfm?id=1855676.1855682

[8] Egele M, Kirda E, Kruegel C. Mitigating drive-by download attacks: Challenges and open problems. In: Camenisch J, Kesdogan D, eds. Proc. of Open Research Problems in Network Security Workshop (iNetSec)|IFIP Advances in Information and Communication Technology. Berlin, Heidelberg: Springer-Verlag, 2009. 52−62. [doi: 10.1007/978-3-642-05437-2_5]

[9] The MITRE Corporation. CVE-2008-6442. 2009. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-6442

[10] The MITRE Corporation. CVE-2007-4105. 2007. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-4105

[11] Microsoft Corporation. Microsoft security bulletin MS06-014—Critical. 2006. http://technet.microsoft.com/en-us/security/bulletin/ms06-014

[12] The MITRE Corporation. CVE-2009-0075, CVE-2010-0249, CVE-2010-0806. 2010. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0075, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-0249, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-0806

[13] Microsoft Corporation. Microsoft security bulletin MS08-078—Critical. 2008. http://technet.microsoft.com/en-us/security/bulletin/MS08-078

[14] Zhang HL, Zhuge JW, Song CY, Zou W. Detection of drive by downloads based on dynamic page views. Journal of Tsinghua University (Science and Technology), 2009,49(S2):2126−2132 (in Chinese with English abstract).

[15] ERCIS. Measurement of drive by download landing sites in the .edu domain (first half year of 2010). Technical Report, 2010 (in Chinese). http://www.edu.cn/scsj_9976/20100907/t20100907_519425.shtml

[16] Wikipedia. Heap spraying. 2008. http://en.wikipedia.org/wiki/Heap_spray

[17] Wikipedia. Address space layout randomization. 2004. http://en.wikipedia.org/wiki/Address_space_layout_randomization

[18] Chen KZ, Gu GF, Zhuge JW, Nazario J, Han XH. WebPatrol: Automated collection and replay of Web-based malware scenarios. In: Proc. of the 6th ACM Symp. on Information, Computer and Communications Security (ASIACCS). New York: ACM Press, 2011. 186−195. [doi: 10.1145/1966913.1966938]

[19] Wikipedia. Document object model. 2001. http://en.wikipedia.org/wiki/Document_Object_Model

[20] Ikinci A, Holz T, Freiling F. Monkey-Spider: Detecting malicious websites with low-interaction honeyclients. In: Proc. of the Sicherheit, Schutz und Zuverlässigkei. 2008. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.210.1385

[21] Seifert C, Welch I, Komisarczuk P. HoneyC—The low-interaction client honeypot. In: Proc. of the 2007 NZCSRCS. 2007. http://www.researchgate.net/publication/202141516_HoneyC_-_The_Low-Interaction_Client_Honeypot

[22] Wang YM. Beck D, Jiang XX, Roussev R, Verbowski C, Chen S, King S. Automated Web patrol with strider honeymonkeys: Finding Web sites that exploit browser vulnerabilities. In: Proc. of the 13th Network and Distributed Systems Security Symp. (NDSS). Reston: The Internet Society (ISOC), 2006. http://www.isoc.org/isoc/conferences/ndss/06/proceedings/papers/honeymonkeys.pdf

[23] Moshchuk A, Bragin T, Gribble SD, Levy HM. A crawler-based study of spyware on the Web. In: Proc. of the 13th Network and Distributed Systems Security Symp. (NDSS). Reston: The Internet Society (ISOC), 2006. http://www.isoc.org/isoc/conferences/ndss/06/proceedings/papers/spycrawler.pdf

[24] CPAN. Mitre honeyclient project. 2007. http://search.cpan.org/~mitrehc/

[25] The Honeynet Project. Capture-HPC. 2008. http://projects.honeynet.org/capture-hpc

[26] Zhang H, Tao R, Li ZY, Du H. The application of judgment matrix approach in detection of vicious script in HTML. Acta Armamentarii, 2008,29(4):469−473 (in Chinese with English abstract).

[27] Zhang H, Tao R, Li ZY, Du H. A vicious script in HTML detection method applied with judgment matrix approach. In: Proc. of the 2007 NetSec. 2007 (in Chinese with English abstract). http://d.g.wanfangdata.com.cn/Conference_6482539.aspx

[28] Seifert C, Welch I, Komisarczuk P. Identification of malicious Web pages with static heuristics. In: Proc. of the Australasian Telecommunication Networks and Applications Conf. (ATNAC). 2008. 91−96. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4783302 [doi: 10.1109/ATNAC.2008.4783302]

[29] Canali D, Cova M, Vigna G, Kruegel C. Prophiler: A fast filter for the large-scale detection of malicious Web pages. In: Proc. of the 20th Int'l World Wide Web Conf. (WWW). New York: ACM Press, 2011. 197−206. [doi: 10.1145/1963405.1963436]

[30] Zhang JJ, Seifert C, Stokes JW, Lee WK. ARROW: GenerAting SignatuRes to detect DRive-by DOWnloads. In: Proc. of the 20th Int'l World Wide Web Conf. (WWW). New York: ACM Press, 2011. 187−196. [doi: 10.1145/1963405.1963435]

[31] Han XH, Gong XR, Zhuge JW, Zou L, Zou W. Detection of drive-by downloads based on the frequent embedded subtree pattern-mining algorithm. Journal of Tsinghua University (Science and Technology), 2011,51(10):1312−1317 (in Chinese with English abstract).

[32] Provos N, Mcnamee D, Mavrommatis P, Wang K, Modadugu N. The ghost in the browser analysis of Web-based malware. In: Proc. of the 1st USENIX Workshop on Hot Topics in Understanding Botnets (HotBots). Berkeley: USENIX Association, 2007. http://dl.acm.org/citation.cfm?id=1323128.1323132

[33] Provos N, Rajab MA, Mavrommatis P. Cybercrime 2.0: When the cloud turns dark. Communications of the ACM, 2009,52(4):42−47. [doi: 10.1145/1498765.1498782]

[34] Wikipedia. Cross-Site scripting. 2003. http://en.wikipedia.org/wiki/Cross-site_scripting

[35] Kieyzun A, Guo PJ, Jayaraman K, Ernst MD. Automatic creation of SQL injection and cross-site scripting attacks. In: Proc. of the 31st Int'l Conf. on Software Engineering (ICSE). Washington: IEEE Computer Society, 2009. 199−209. [doi: 10.1109/ICSE.2009.5070521]

[36] Wang T, Yu SZ, Xie BL. A novel framework for learning to detect malicious Web pages. In: Proc. of the 2010 Int'l Forum on Information Technology and Applications (IFITA). Washington: IEEE Computer Society, 2010. 353−357. [doi: 10.1109/IFITA.2010.173]

[37] Louw MT, Venkatakrishnan VN. Blueprint: Robust prevention of cross-site scripting attacks for existing browsers. In: Proc. of the 30th IEEE Symp. on Security and Privacy (S&P). Washington: IEEE Computer Society, 2009. 331−346. [doi: 10.1109/SP.2009.33]

[38] Su ZD, Wassermann G. The essence of command injection attacks in Web applications. In: Proc. of the 33rd ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL). New York: ACM Press, 2006. 372−382. [doi: 10.1145/1111320.1111070]

[39] Moshchuk A, Bragin T, Deville D, Gribble SD, Levy HM. SpyProxy: Execution-Based detection of malicious Web content. In: Proc. of the 16th USENIX Security Symp. Berkeley: USENIX Association, 2007. 27−42. http://dl.acm.org/citation.cfm?id=1362903.1362906

[40] Rieck K, Krueger T, Dewald A. Cujo: Efficient detection and prevention of drive-by-download attacks. In: Proc. of the 26th Annual Computer Security Applications Conf. (ACSAC). New York: ACM Press, 2010. 31−39. [doi: 10.1145/1920261.1920267]

[41] Reis C, Dunagan J, Wang HJ, Dubrovsky O, Esmeir S. BrowserShield: Vulnerability-Driven filtering of dynamic HTML. In: Proc. of the 7th USENIX Symp. on Operating Systems Design and Implementation (OSDI). Berkeley: USENIX Association, 2006. 61−74. http://dl.acm.org/citation.cfm?id=1298455.1298462

[42] Li Z, Yi T, Cao YZ, Rastogi V, Chen Y, Liu B, Sbisa C. WebShield: Enabling various Web defense techniques without client side modifications. In: Proc. of the 18th Network & Distributed System Security Symp. (NDSS). Reston: The Internet Society (ISOC), 2011. http://www.isoc.org/isoc/conferences/ndss/11/pdf/6_2.pdf

[43] Seifert C, Delwadia V, Komisarczuk P, Stirling D, Welch I. Measurement study on malicious Web servers in the .nz domain. In: Boyd C, Nieto JG, eds. Proc. of the 14th Australasian Conf. on Information Security and Privacy (ACISP). Berlin, Heidelberg: Springer-Verlag, 2009. 8−25. [doi: 10.1007/978-3-642-02620-1_2]

[44] Egele M, Wurzinger P, Kruegel C, Kirda E. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In: Flegel U, Bruschi D, eds. Proc. of the 6th Conf. on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA). Berlin, Heidelberg: Springer-Verlag, 2009. 88−106. [doi: 10.1007/978-3-642-02918-9_6]

[45] Mozilla. SpiderMonkey—MDC Docs. 2012. https://developer.mozilla.org/en/SpiderMonkey

[46] libemu—x86 shellcode emulation. 2007. http://libemu.carnivore.it/

[47] Ratanaworabhan P, Livshits B, Zorn B. NOZZLE: A defense against heap-spraying code injection attacks. In: Proc. of the 18th USENIX Security Symp. Berkeley: USENIX Association, 2009. 169−186. http://dl.acm.org/citation.cfm?id=1855768.1855779

[48] Ding Y, Wei T, Wang TL, Liang ZK, Zou W. Heap taichi: Exploiting memory allocation granularity in heap-spraying attacks. In: Proc. of the 26th Annual Computer Security Applications Conf. (ACSAC). New York: ACM Press, 2010. 327−336. [doi: 10.1145/1920261.1920310]

[49] Curtsinger C, Livshits B, Zorn B, Seifert C. Zozzle: Fast and precise in-browser JavaScript malware detection. In: Proc. of the 20th USENIX Security Symp. Berkeley: USENIX Association, 2011. http://dl.acm.org/citation.cfm?id=2028067.2028070

[50] Song CY, Zhuge JW, Han XH, Ye ZY. Preventing drive-by download via inter-module communication monitoring. In: Proc. of the 5th ACM Symp. on Information, Computer and Communications Security (ASIACCS). New York: ACM Press, 2010. 124−134. [doi: 10.1145/1755688.1755705]

[51] Lu L, Yegneswaran V, Porras P, Lee WK. BLADE: An attack-agnostic approach for preventing drive-by malware infections. In: Proc. of the 17th ACM Conf. on Computer and Communications Security (CCS). New York: ACM Press, 2010. 440−450. [doi: 10.1145/1866307.1866356]

[52] Wang HJ, Grier C, Moshchuk A, King ST, Choudhury P, Venter H. The multi-principal OS construction of the gazelle Web browser. In: Proc. of the 18th Usenix Security Symp. (USENIX Sec). Berkeley: USENIX Association, 2009. 417−432. http://dl.acm.org/citation.cfm?id=1855768.1855794

[53] Wang HJ, Moshchuk A, Bush A. Convergence of desktop and Web applications on a multi-service OS. In: Proc. of the 4th USENIX Workshop on Hot Topics in Security. Berkeley: USENIX Association, 2009. http://dl.acm.org/citation.cfm?id=1855628.1855639

[54] Wang HJ, Fan X, Howell J, Jackson C. Protection and communication abstractions for Web browsers in MashupOS. In: Proc. of the 21st ACM Symp. on Operating Systems Principles (SOSP). New York: ACM Press, 2007. 1−16. [doi: 10.1145/1294261.1294263]

[55] Yee B, Sehr D, Dardyk G, Chen JB, Muth R, Ormandy T, Okasaka S, Narula N, Fullagar N. Native client: A sandbox for portable, untrusted x86 native code. In: Proc. of the 30th IEEE Symp. on Security and Privacy (S&P). Washington: IEEE Computer Society, 2009. 79−93. [doi: 10.1109/SP.2009.25]

[56] MSDN. IE8 security part V: Comprehensive protection. 2008. http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-protection.aspx

[57] VUPEN Security. VUPEN vulnerability research demo/VUPEN 0-day exploit for Google chrome (Sandbox/ASLR/DEP Bypass). 2011. http://www.vupen.com/demos/VUPEN_Pwning_Chrome.php

[58] Drivesploit Project. Drivesploit. 2010. http://blog.armorize.com/2010/07/drivesploit-source-code-released.html

[59] Grier C, Thomas K, Paxson V, Zhang M. @spam: The underground on 140 characters or less. In: Proc. of the 17th ACM Conf. on Computer and Communications Security (CCS). New York: ACM Press, 2010. 27−37. [doi: 10.1145/1866307.1866311]

[60] Symantec Corporation. Symantec announces february 2011 MessageLabs intelligence report. 2011. http://www.symantec.com/about/news/release/article.jsp?prid=20110301_01

[61] Tzermias Z, Sykiotakis G, Polychronakis M, Markatos EP. Combining static and dynamic analysis for the detection of malicious documents. In: Proc. of the 2011 European Workshop on System Security (EUROSEC). New York: ACM Press, 2011. [doi: 10.1145/ 1972551.1972555]

[62] Ford S, Cova M, Kruegel C, Vigna G. Analyzing and detecting malicious flash advertisements. In: Proc. of the 25th Annual Computer Security Applications Conf. (ACSAC). New York: ACM Press, 2009. 363−372. [doi: 10.1109/ACSAC.2009.41]

[63] Zdnet. Pwn2Own 2010: iPhone hacked, SMS database hijacked|ZDNet. 2010. http://www.zdnet.com/blog/security/pwn2own-2010-iphone-hacked-sms-database-hijacked/5836

[64] Exploit-db. Android 2.0-2.1 reverse shell exploit. 2010. http://www.exploit-db.com/exploits/15423/

附中文参考文献:

[14] 张慧琳,诸葛建伟,宋程昱,邹维.基于网页动态视图的网页木马检测方法.清华大学学报(自然科学版),2009,49(S2):2126−2132.

[15] 北京大学网络与信息安全实验室.2010 年上半年教育网网站挂马监测分析报告.2010. http://www.edu.cn/scsj_9976/20100907/t20100907_519425.shtml

[26] 张昊,陶然,李志勇,杜华.判断矩阵法在网页恶意脚本检测中的应用.兵工学报,2008,29(4):469−473.

[27] 张昊,陶然,李志勇,杜华.网页恶意脚本检测方法研究.见:全国网络与信息安全技术研讨会文集.2007. http://d.g.wanfangdata.com.cn/Conference_6482539.aspx

[31] 韩心慧,龚晓锐,诸葛建伟,邹磊,邹维.基于频繁子树挖掘算法的网页木马检测技术.清华大学学报(自然科学版),2011,51(10):1312−1317.

张慧琳(1987－),女,河南淮阳人,博士生,主要研究领域为恶意代码检测与防范,互联网安全监测.
E-mail: zhanghuilin@pku.edu.cn

韩心慧(1969－),男,博士,高级工程师,CCF会员,主要研究领域为恶意代码检测与防范,网络安全监测技术.
E-mail: hanxinhui@pku.edu.cn

邹维(1964－),男,研究员,博士生导师,CCF高级会员,主要研究领域为互联网安全检测,软件安全.
E-mail: zouwei_64@hotmail.com