

基于余弦相似计算原理的 Webshell 检测方法

卢冬海, 王洋洋, 杨春艳

(宁波大学 图书馆与信息中心, 浙江 宁波 315211)

摘要: 常规 Webshell 检测方法一般基于代码特征库和基于通信特征. 代码混淆和虚假通信等反检测技术的出现, 提升了 Webshell 隐蔽度, 常规方法难于发现. 为了检测此类隐蔽度较高的 Webshell, 采用了抽取程序代码多维 Webshell 特征, 用余弦相似计算原理度量代码文件间的相似度, 平均相似度极低的文件即为可疑 Webshell 文件. 实验表明, 正常代码文件间的平均相似度在 $10^{-1} \sim 10^{-2}$ 量级, 而 Webshell 文件和正常代码文件的平均相似度在 10^{-4} 量级, 此方法可以有效地检测出 Webshell.

关键词: Webshell; 相似度; 余弦定理; 网站安全

中图分类号: TN915.08

文献标志码: A

文章编号: 1001-5132 (2017) 06-0012-04

B/S 架构的 Web 系统拥有便捷的部署和使用特性, 因此被广泛地应用. 随之而来, Web 系统的安全问题日益凸显, 如网页被篡改、数据被窃取、服务器被入侵等等. 入侵者成功入侵后, 通常在 Web 服务器留下网站后门, 即 Webshell. 通过 Webshell 获取对网站甚至服务器的操作权限, 达到非法目的. 入侵者甚至利用大量的 Webshell 组成僵尸网络去攻击新的目标^[1]. 奇虎 360 公司在《2015 年中国网站安全报告》中指出: 2015 年 360 网站对 21 854 台网站服务器进行网站后门检测, 覆盖网站 322.3 万个, 扫描发现 4 097 台服务器存在后门, 占扫描网站服务器的 18.7%. 2015 年已扫出各类网站后门文件样本多达 858.1 万个^[2].

网站被植入 Webshell, 意味着入侵者留下了一扇可以随时进出网站服务器的后门, 普通网站管理员很难发现这个后门, Webshell 危害程度极大. 因此, 如何有效地检测出 Webshell 对于 Web 系统的安全有着极其重要的意义. 当前, 业界检测的方法主要有 2 种: 基于代码特征库检测和基于通信特征检测. 这 2 种方法可以检测出大部分的 Webshell, 而对采用了代码混淆和虚假通信技术的 Webshell 的检测效果却不理想.

本文通过比较程序文件的相似度, 寻找孤立

的文件, 可有效地检测出采用了代码混淆和虚假通信技术的 Webshell.

1 相关研究

1.1 Webshell 的相关特征

Webshell 顾名思义, Web 表示需要服务器开放 Web 服务, shell 表示取得对服务器某种程度上操作权限. 它是通过 Http 的访问方式, 给服务器发送操作指令, 可以穿透防火墙, 其行为特征和正常的 Web 访问几乎一致. 传统的防火墙和杀毒软件较难发现, 也不会系统在安全日志中留下痕迹, 只会 Web 日志中留下和正常 Web 访问日志混在一起的日志记录, 因而普通的网站管理员很难发现.

Webshell 出现形态通常以脚本文件的形式, 例如 asp、php、jsp、py 等. 其产生途径一般是入侵者在有上传文件功能的 Web 系统中利用系统漏洞上传到服务器. Web 系统一般有上传图片、word 文档、压缩文件等功能. 如上传功能的程序代码有漏洞, 未对或不能对上传的文件做严格的检测, 入侵者便可利用此漏洞, 上传 Webshell 到服务器^[3]. 按照功能, 一般分为 3 类: 大马、小马及一句话木马.

大马: 拥有丰富的操作接口, 友好的图形界面, 集成了操作文件、操作数据库、执行系统命令等强

大功能,其体积较大、文件数量和代码量较多,代码通常采用混淆技术以躲避查杀软件。由于其拥有友好的操作界面,使用大马的用户不需要具有很好的计算机知识,通过鼠标点击即可完成相应的操作,不必输入复杂的命令。

小马:一般只拥有单一的功能,例如获取数据库或下载文件等。一般为单个文件,代码量较少,文件大小通常少于30字节。使用者需要具备较好的计算机知识。通过小马作为跳板,可以把小马“养”成大马。

一句话木马:指木马文件只包含一句代码。这句代码的功能是执行用户精心构造的命令字符串,字符串通过http协议的post或get方法提交到服务器。理论上可以构造任何命令,但服务器通常会转义“.”“%”“(”等特殊字符,这给入侵者造成很多麻烦。入侵者需要精心构造语句,绕过或避免特殊字符被转义。例如php的一句话木马原型 $\text{eval}(\$_GET['str'])$ ^[4]。这一句便可给网站留下后门。一句话木马也是被使用最多的木马。

1.2 基于代码特征库检测

通过收集网络上存在的Webshell,提取相关特征,进而形成特征库;然后根据特征库扫描代码文件,发现木马。Emposha开发的Web shell detector使用了一个较大的Webshell特征库,可以用来识别php/cgi/asp/aspx的Webshell,识别率高达99%^[5]。D盾_IIS防火墙可以根据特征库识别一句话木马以及采用混淆技术的木马^[6]。显然,根据特征库检测方法的关键在于特征库的质量。特征库需要及时更新,否则无法检测出新出现的木马。更新必会有滞后性,这也是根据特征库检测方法的一个重要缺陷。

1.3 基于通信特征检测

Webshell一般被入侵者控制,普通用户只会访问正常的网页文件。Webshell文件的访问频率明显不同于其他文件的访问频率。访问Webshell文件的http数据包中的payload也有别于其他正常访问。访问Webshell文件的数据包中的payload通常包含一些可执行的命令文本,并且payload长度较长。访问正常网页文件的http数据包中的payload一般是有意义的参数。例如id=123、category=news, payload的长度也较短,我们统称这些为通信特征。

利用统计学、大数据分析的方法识别出异常的通信特征,找出Webshell。石刘洋等^[7]已提出了通过对服务器日志文本文件的分析,从文本特征、统计特征和页面关联特征3个角度检测Webshell。

2 基于余弦相似计算原理的检测方法

程序代码中有程序员自定义的变量、函数、类,这些自定义元素会在其他地方被调用,整个软件代码就是由这些自定义元素和程序语言本身提供的API组成。程序代码有一定的内在逻辑,程序员对自定义元素的命名有自己的命名习惯,对于函数getNameById,不同的程序员有不同的命名,例如GetNameById、GetNameByID、get_name_by_id。

正常程序文件的自定义元素的命名方式与Webshell程序的命名方式明显不同。同时,这些自定义函数、类和变量通常只出现在正常的程序文件中,不会出现在Webshell文件。Webshell程序代码因其独特的功能,需要使用一些特殊的API,例如执行系统级命令、上传文件、删除文件等,我们统称其为Webshell特征元素。Webshell特征元素在正常的程序代码中极少使用。所以,正常的程序文件之间有显著的相似度,而与Webshell文件的相似度较低。利用程序本身自定义元素和Webshell程序中常使用的API(exec, eval, upload等)作为表示文件的向量维度,组成一个向量(下文统称为文件向量),根据余弦定理计算出文件间的相似度,最后以平均相似度来衡量文件是否为孤立文件,即可疑Webshell。

2.1 文件向量的定义

文件向量由自定义元素和Webshell特征元素组成,我们用 $V(x_1, x_2, x_3, \dots, x_m, y_1, y_2, y_3, \dots, y_n)$ 表示 $(m+n)$ 维的文件向量,其中 x_m 表示自定义元素维度, y_n 表示Webshell特征元素维度。

$$x_m = F(x_m)L(x_m)P1(x_m),$$

其中: $F(x_m)$ 为 x_m 在文件中出现的频度; $L(x_m)$ 为 x_m 名字的字符数(字符数越多,区别度越大); $P1(x_m)$ 为 x_m 分别为普通变量名、函数名、版本信息时的权值,详见表1。

$$y_n = F(y_n)P2(y_n),$$

其中: $F(y_n)$ 为 y_n 在文件中出现的频度; $P2(y_n)$ 为Webshell不同特征元素时的权值,详见表2。

表 1 自定义元素权值

自定义元素	$P1(x_n)$
变量	4
函数	5
类	8
版本信息	12

表 2 Webshell 特征元素权值

Webshell 特征元素	$P2(y_n)$
exec	10
eval	10
fopen	5
fputs	5
copy	4

2.2 相似度计算

两个向量相似度的余弦定理计算公式为:

$$\cos \theta = \frac{\langle \mathbf{A}, \mathbf{B} \rangle}{(\|\mathbf{A}\| \|\mathbf{B}\|)} = \frac{x_1y_1 + x_2y_2 + \cdots + x_ny_n}{\sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \sqrt{y_1^2 + y_2^2 + \cdots + y_n^2}} \quad (1)$$

向量 $\mathbf{A}(x_1, x_2, \cdots, x_n)$ 和向量 $\mathbf{B}(y_1, y_2, \cdots, y_n)$ 的夹角余弦值即为其相似度, 余弦值越小表明相似度越大, 越大则相似度越小. 由于向量中的每一个变量都是正数, 因此余弦的取值在 0~1 之间, 即夹角在 $0^\circ \sim 90^\circ$ 之间. 当 2 个文件向量的余弦值为 1 时, 这 2 个向量的夹角为 0, 2 个文件完全相同. 当夹角余弦接近 1 时, 2 个文件越相似; 相反, 当夹角余弦接近 0 时, 2 个文件差异越大^[8]. 为了减少误报概率, 采用平均相似度来比较.

设有文件 $f_1, f_2, \cdots, f_m$, F_i 为 f_i 的文件向量, 则 f_i 的平均相似度计算公式为:

$$A(f_i) = \sum_{j=1}^m \cos \langle F_i, F_j \rangle / C_m^2, \quad (2)$$

其中: $\cos \langle F_i, F_j \rangle$ 表示向量 F_i 和 F_j 的余弦值, C_m^2 表示在 m 个文件中任取 2 个的组合数量.

通过比较文件的平均相似度, 发现 Webshell 文件的平均相似度远低于正常文件. 正常文件的平均相似度一般在 $10^{-1} \sim 10^{-2}$ 数量级, Webshell 文件的平均相似度一般在 10^{-4} 数量级.

3 实验验证

选取流行的内容管理系统 Dedecms 作为实验对象. Dedecms 是以 php 为程序语言开发的网站内

容管理系统, 在国内被广泛使用. 选取 8 个 php 的 Webshell 文件, 进行如下实验验证:

(1)提取自定义元素, 统计自定义元素在全部程序中的使用频率. php 语言中的自定义变量是以 \$ 符号开头与以字母或下划线组成, 函数定义的形式为 function name, 类定义形式为 class name. 根据这些特征, 提取自定义元素. 为了减少计算的复杂度, 选取使用频率最高的 1 000 个自定义元素作为文件向量的维度, 详见表 3.

表 3 自定义元素出现频率

排序	元素名称	频率/次	排序	元素名称	频率/次
1	\$row	3 543	16	\$str	457
2	\$dsq	2 338	17	\$dlist	455
3	ShowMsg	1 067	18	\$channelid	432
4	\$id	907	19	\$sql	408
5	\$ctag	850	20	\$msg	391
6	\$aid	762	21	\$arr	389
7	\$fp	691	22	\$dtp	347
8	\$dopost	656	23	\$action	345
9	\$typeid	633	24	\$key	329
10	\$filename	597	25	\$val	328
11	\$win	582	26	\$keyword	301
12	\$rs	579	27	typename	298
13	\$query	578	28	\$title	292
14	\$id:	471	29	\$vs	288
15	\$cfg_ml	464	...	...	...

(2)形成文件向量. 表 3 的元素和 Webshell 特征元素作为文件向量的维度. 对于一些被其他文件包含的文件, 不计算其文件相似度, 而是把被包含文件的代码合并到包含文件的文件中计算.

(3)计算平均相似度. 根据式(1)和式(2)得出文件的平均相似度(表 4).

从表 4 可发现, Webshell 文件的平均相似度(通常在 10^{-4} 数量级)明显低于正常文件的相似度(一般在 $10^{-1} \sim 10^{-2}$ 数量级). 再结合是否存在 Webshell 特征元素, 可确认其是否为 Webshell 文件.

表 4 文件平均相似度及排序

文件	平均相似度	排序	备注
phpqrcode.php	0.360 435 121 2	1	正常文件
sphinxclient.class.php	0.307 415 128 2	2	正常文件
arc.archives.class.php	0.288 187 236 4	3	正常文件

续表			
文件	平均相似度	排序	备注
Lurd.class.php	0.281 352 699 8	4	正常文件
dedecollection.class.php	0.272 332 976 7	5	正常文件
arc.listview.class.php	0.250 764 413 8	6	正常文件
dedetag.class.php	0.191 817 543 6	7	正常文件
arc.sglstview.class.php	0.191 087 647 3	8	正常文件
...	...	...	...
_info.php	0.000 070 493 9	537	Webshell
p.php	0.000 064 190 2	538	Webshell
Default_.php	0.000 062 001 2	539	Webshell
D_shell.php	0.000 060 611 6	540	Webshell
Index_.php	0.000 055 895 8	541	Webshell
post3.php	0.000 053 456 8	542	Webshell
get.php	0.000 037 577 4	543	Webshell
eval.php	0.000 004 114 8	544	Webshell

4 结语

基于余弦相似度计算原理, 提出了计算代码相似度方法来检测 Webshell. 该方法依据程序员开发 Web 系统的命名习惯有别于 Webshell 制造者习惯以及 Web 系统本身的内在代码逻辑有别于

Webshell 代码, 抽取出代表文件向量的维度, 结合 Webshell 的特征维度组成一个代表文件的文件向量, 再利用余弦定理计算出相似度. 以平均相似度来判别代码文件是否为孤立文件, 即可疑 Webshell 文件. 实验表明, 该方法可有效地用于 Webshell 的检测.

参考文献:

[1] 李可, 方滨兴, 崔翔, 等. 基于 Webshell 的僵尸网络研究[J]. 通信学报, 2016, 37(6):12-18.

[2] 360 互联网安全中心. 奇虎 2015 网站安全报告[EB/OL]. [2016-10-12]. <http://zt.360.cn/1101061855.php?dtid=1101062368&did=1101536490>.

[3] Netcraft. Web server survey[EB/OL]. [2016-08-16]. <https://news.netcraft.com/archives/2015/11/16/november-2015-web-server-survey.html>.

[4] F5. F5 Technologies[EB/OL]. [2016-08-03]. <https://devcentral.f5.com/articles/webshells>.

[5] THORNTON J. Web shell detector[EB/OL]. [2016-08-12]. <https://github.com/emposha/PHP-Shell-Detector>.

[6] 杨明, 王轶骏, 薛质. 高混淆网页木马的研究与检测实现[J]. 计算机工程与设计, 2004, 35(8):2634-2642.

[7] 石刘洋, 方勇. 基于 Web 日志的 Webshell 检测方法研究[J]. 信息安全研究, 2016, 1(2):66-71.

[8] 吴军. 数学之美[M]. 北京: 人民邮电出版社, 2014: 127-130.

A method of detecting Webshell based on cosine similarity calculation principle

LU Dong-hai, WANG Yang-yang, YANG Chun-yan
(Library and Information Center, Ningbo University, Ningbo 315211, China)

Abstract: The conventional Webshell detection method is based on code feature library or on the characteristics of communication. With the hidden degree increasing, the difficulty with Webshell detection grows using the conventional method. In order to detect the Webshell that has higher hidden degrees, we make multidimensional vectors of Webshell feature from the code files, then compare the similarity between code files using cosine similarity calculation principle. The code file which has very low average similarity is considered a good candidate for Webshell file. Experiment shows that the average similarity between the normal code files is in order of magnitude ranging from 10^{-1} to 10^{-2} , whereas the average similarity between the normal code file and Webshell file is in the range of 10^{-4} . Therefore, this method can effectively detect Webshell.

Key words: Webshell; similarity; cosine theorem; web security

(责任编辑 史小丽)