ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.4, April 2010, pp.816−826
doi: 10.3724/SP.J.1001.2010.03515

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

基于阳性选择的蠕虫检测系统*

洪 征+, 吴礼发

(解放军理工大学 指挥自动化学院 计算机系,江苏 南京 210007)

Worm Detection System Based on Positive Selection

HONG Zheng+, WU Li-Fa

(Department of Computer Science, Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)

+ Corresponding author: E-mail: hongzhengjs@139.com

Hong Z, Wu LF. Worm detection system based on positive selection. *Journal of Software*, 2010,21(4):816−826.
http://www.jos.org.cn/1000-9825/3515.htm

Abstract: Worms search for targets by means of service requests, and anomalous service requests give indication of worm propagation. A worm detection system that uses positive selection algorithm to characterize normal service requests with self-strings is proposed. Bloom filters are used to represent hosts' self-strings and monitor the network for suspicious service requests. On the basis of worm properties, the discovered suspicious service requests are correlated in the form of binary trees, and a non-parametric CUSUM (cumulative sum) algorithm is used to monitor the anomaly value of binary trees so as to detect worm propagation timely and accurately. Experimental results of the GTNetS (Georgia Tech Network Simulation) platform show that the proposed system is effective to detect worms, and the system's influence on normal network traffic is minor.

Key words: worm; artificial immune system; positive selection; binary tree; network simulation

摘 要: 蠕虫通过发送网络服务请求搜寻感染目标,主机的异常网络服务请求可以作为蠕虫检测的依据.提出了一种蠕虫检测系统,基于阳性选择算法构造自体字符串集合描述主机的正常网络行为.自体字符串集合采用 Bloom filter过滤器的形式表示,用于监视主机的网络行为以发现网络中可疑的网络服务请求.依据蠕虫的传播特征,采用二叉树的形式对所发现的可疑网络服务请求进行关联分析,通过无参 CUSUM(cumulative sum)算法监视二叉树异常值的变化,从而及时、准确地发现蠕虫传播.GTNetS(Georgia Tech Network Simulation)平台的测试实验结果表明,所提出的蠕虫检测系统能够有效检测蠕虫,同时对于主机正常网络通信的影响较小.

关键词: 蠕虫;人工免疫系统;阳性选择;二叉树;网络仿真
中图法分类号: TP393 **文献标识码**: A

按照文伟平对蠕虫的定义[1],蠕虫通过扫描和攻击网络上存在系统漏洞的节点主机,在网络上扩散.该定义是狭义的蠕虫定义,所涉及的蠕虫是主动探测蠕虫.近年来,E-mail 蠕虫、即时通信蠕虫[2]开始在网络上出现.但

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2004AA147070 (国家高技术研究发展计划(863))

Received 2007-11-09; Revised 2008-05-19; Accepted 2008-10-27

是,以 CodeRed,Blaster 为代表的主动探测蠕虫仍然是网络安全的重要威胁.本文以主动探测蠕虫为研究对象,并在文中将其简称为蠕虫.

作为一种完全自动的网络攻击形式[3],蠕虫有一些难以隐匿的特征,可以作为蠕虫检测的依据.首先,在蠕虫出现时,网络上往往有大量相似的网络服务请求.其次,蠕虫传播具有因果联系.被感染的主机会出现与感染其主机相似的异常网络行为.另外,蠕虫传播往往会产生大量无效服务请求,表现为 ICMP-T3 主机不可达消息以及 TCP_RST 消息的大量出现.

本文提出了一种蠕虫检测系统,基于主机的网络服务请求信息发现网络中的蠕虫攻击.检测系统采用阳性选择算法生成自体字符串集合.自体字符串集合以 Bloom filter 过滤器的形式构建,监视主机的网络行为以及时发现主机的可疑网络服务请求.对于所发现的可疑网络服务请求,将依据蠕虫的传播特征,采用二叉树的结构进行关联分析,通过无参 CUSUM(cumulative sum)算法判定网络中是否存在蠕虫攻击.

1 应用阳性选择算法检测蠕虫

蠕虫以网络服务请求的形式在网络上搜寻潜在的感染目标.一台网络主机在正常工作的情况下所请求的网络服务的种类往往非常有限,而涉及的目标主机也相对固定.在主机感染蠕虫以后,由于蠕虫通常会积极扩散,主机作为蠕虫节点会发出大量的网络服务请求搜寻攻击目标,此时,主机发出的很多网络服务请求在主机正常工作时并不出现.因此,发出异常的网络服务请求是主机感染蠕虫的重要信号,可以作为蠕虫检测的依据.问题的关键在于如何高效地监控主机的网络服务请求信息.

1.1 阴性选择算法的分析

在生物免疫系统中,阴性选择机制对淋巴细胞进行筛选,保证成熟的淋巴细胞能够识别外部抗原,同时不会攻击体内正常细胞.受阴性选择机制启发,Forrest[4]提出了阴性检测算法.阴性选择算法在自体样本的基础上,遵从一定的匹配规则产生足够数量的检测器建立检测器集合.此后,利用检测器集合对系统进行监视以发现异常.

检测器集合是检测系统的核心,检测器集合的生成对于检测系统意义重大.Forrest 采用穷举法生成字符串形式的检测器集合,生成检测器集合的时间复杂度为 $O\left(\frac{-\ln(P_f)}{P_m\times(1-P_m)^{N_S}}\times N_S\right)$,其中:检测器集合的漏报率以 P_f 表示,P_m 表示两个完全随机的字符串相匹配的概率;N_S 表示自体样本集合的大小,即采用穷举法,生成检测器集合的时间复杂度是 N_s 的指数级.对于蠕虫检测系统,自体样本是主机在正常工作情况下发送的网络服务请求.一台主机的正常网络服务请求虽然有限,但指数级的时间复杂度将使生成检测器集合的时间过长.

D'haeseleer[5]提出了在以 r-连续位匹配作为匹配规则的情况下,两种检测器集合的生成算法.第 1 种算法通过模板对候选字符串进行预先处理,采用这种算法生成检测器集合的时间复杂度为 $O((l-r)\times N_S)+O((l-r)\times 2^r)+O(l\times N_R)$,空间复杂度为 $O((l-r)^2\times 2^r)$.其中,l 代表字符串的长度,r 代表连续位匹配的匹配阈值,N_S 表示自体样本集合的大小,N_R 代表生成的检测器集合大小.所提出的另外一种算法被称为贪婪算法,通过避免检测字符串之间相互重叠所造成的冗余,保证检测器集合最广泛地覆盖非自体空间.算法产生检测器集合的时间复杂度为 $O((l-r)\times 2^r\times N_R)$,空间复杂度为 $O((l-r)^2\times 2^r)$.两种算法产生检测器集合的时间复杂度以及空间复杂度都是 r 的指数级,即在 r 值较高时,算法的时、空开销都很大.如果要保证检测器集合的漏报率最低,$N_R\propto 2^r\times(1-2^{-r})^{N_S}$,即 N_R 是 N_S 的指数级,在这种情况下,生成检测器集合的时间复杂度为 N_S 的指数级,等同于穷举法生成检测器集合的时间复杂度.

Stibor[6]针对 r-chunk 匹配算法提出了一种检测器集合生成算法,算法以散列表的形式表示由 r-chunk 字符串及检测位置组成的检测器.算法生成检测器集合的时间复杂度为 $O((l-r)\times 2^r)+O(N_S\times(l-r))+O(2^r)$,时间复杂度是参数 r 的指数级,因此只适用于 r 值较小的情况.蠕虫检测系统需要以二进制字符串表示主机的网络服务请求.监视主机的网络通信,如果限定只能采用较小的匹配 r 值,将导致大量的检测漏洞.因为 r 值较小意味着检测器具有很强的概括能力,对于一些非自体空间,能够覆盖这些区域的检测器按照匹配规则可能同时匹配一些自

体样本,这些检测器将在阴性选择的过程中被删除,直接结果是无法生成检测器对很多非自体区域形成覆盖,导致检测漏洞的大量存在.

上述算法均为字符串形式的阴性选择算法.此外,针对实值形式的问题,González[7]提出以实值形式表示自体、非自体空间,采用欧几里德距离描述检测器对周围数据点的覆盖.Ji[8]提出了检测半径可变的实值阴性选择算法 V-detector,算法产生的检测器集合能够高效、充分地覆盖非自体空间.

针对所考虑的蠕虫检测问题,采用字符串形式的阴性选择算法生成检测器集合,存在计算开销、存储开销过大的问题.而实值形式的阴性选择算法由于表示空间的差异,也不适合直接使用.

1.2 阳性选择算法

在免疫系统中除了阴性选择,还有阳性选择(positive selection)机制.T 淋巴细胞如果能够识别 MHC 复合体,阳性选择可以保证这些对机体有益的 T 淋巴细胞能够被保留下来.

针对所考虑的监视主机网络通信的问题,提出了如下的阳性选择算法生成自体字符串集合:

(1) 监视主机的网络通信,提取主机发出的网络服务请求信息;

(2) 以网络服务请求信息中的一些字段为基础构造自体字符串 s_i;

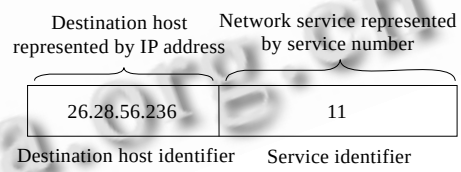
(3) 判断自体字符串 s_i 是否已经在自体字符串集合 S 当中,如果不存在,则将 s_i 加入集合 S;

(4) 在训练完成以后,将得到一个描述主机正常网络行为的自体字符串集合 S,利用集合 S 对主机的网络行为进行监视.对于主机发出的网络服务请求,通过查询在集合 S 中是否有对应项,判断该网络服务请求属于正常的网络服务请求,还是属于可疑的网络服务请求.

自体字符串被用于描述主机的正常网络服务请求.通过构造映射表,将(端口,协议)组合所标识的网络服务映射为服务序号,如图 1(a)所示.自体字符串由目的主机标识以及由服务序号表示的目标服务两部分构成,如图 1(b)所示.

Service number	Port	Protocol	Network service
0	7	TCP	Echo
1	7	UDP	Echo
2	9	TCP	Discard
3	9	UDP	Discard
4	13	TCP	Daytime
5	13	UDP	Daytime
…	…	…	…

(a) Mapping from a (port, protocol) combination to a service number
(a) (端口,协议)组合映射为服务序号



(b) Structure of a self-string
(b) 自体字符串的结构

Fig.1 Representation of self-strings

图 1 自体字符串的表示

在阳性选择算法中,自体字符串充当自体检测器的角色,自体字符串集合被用于实现对自体空间的覆盖.Bloom filter[9]是表示自体字符串集合的理想形式,采用 Bloom filter 表示自体字符串集合无须存储具体的自体字符串信息,能够大幅度减少系统需要的存储空间;而且对 Bloom filter 进行隶属判定的时间复杂度为 $O(1)$,这使自体字符串集合可以应用于在线检测.

Bloom filter 通过一组散列函数及相应的字符串测试一个对象是否隶属于特定集合.一个 Bloom filter 需要使用 k 个值域为[1,m]的、相互独立的散列函数:$h_1,h_2,h_3,\ldots,h_k$,同时需要一个长度为 m 的字符串对应于这组散列函数.

假设进行隶属关系判定的集合 M 包含 n 个对象,$M=\{x_1,x_2,x_3,\ldots,x_n\}$.若进行隶属关系的测试,则 Bloom filter

首先用散列函数集合以及相应的字符串对集合 M 进行特征化,该过程分为两步:

(1) 将字符串中的所有位初始化为 0;

(2) 对于集合 M 中的每一个对象,分别使用 k 个散列函数进行计算,并根据计算结果将字符串的相应位设置为 1.例如,如果对象 x_1 对应于散列函数 h_2 的结果是 $h_2(x_1)=3$,则将字符串的第 3 位设置为 1.

在完成以上步骤之后,可以利用散列函数集合和相应的字符串来判断一个对象是否隶属于集合 M.以 y_1 表示待判定的对象,应用 k 个散列函数对其进行计算:$h_1(y_1),h_2(y_1),h_3(y_1),\ldots,h_k(y_1)$.如果对于 $1\le i\le k$,字符串的第 $h_i(y_1)$ 位都为 1,则推断 y_1 隶属于集合 M;如果存在 $h_i(y_1)$,字符串的第 $h_i(y_1)$ 位不是 1,则推断 y_1 不属于集合 M.

如果对于 $x\notin M$,对应于 k 个散列函数的字符串在 $h_1(x),h_2(x),h_3(x),\ldots,h_k(x)$ 的位置上恰巧都为 1,Bloom filter 将发生误报.Bloom filter 的误报问题无法根本消除,但若设定 $k=\frac{m}{n}\ln 2$,Bloom filter 的误报率 f 将降至最低[10],此时 $f=2^{-k}$.

为了建立主机的自体字符串集合,蠕虫检测系统需要有一段训练时间.在此阶段,系统收集主机的正常网络服务请求信息建立自体字符串集合,并以 Bloom filter 的形式表示.此后,系统将利用自体字符串集合对主机发出的网络服务请求进行判定,如果主机发出的网络服务请求隶属于自体字符串集合,则允许发送,否则将被作为可疑的网络服务请求作进一步的判定处理.

1.3 对自体字符串集合的讨论

按照阳性选择算法,检测系统通过训练,依据主机的正常网络通信情况建立自体字符串集合.在系统的训练阶段,由于网络环境的复杂性,尽管主机发出的绝大部分服务请求是正常的,但其中也可能混杂着一些具有恶意的或者说异常的服务请求.自体字符串集合生成以后,将无法发现在训练阶段出现过的异常网络服务请求,这将导致漏报,即把异常网络服务请求判定为正常.另外,Bloom filter 在判定时也存在一定的误报率,可能将少数可疑网络服务请求判定为正常.

这些漏报对于系统的检测性能不会产生很大的影响.蠕虫的传播往往会产生大量的异常网络服务请求,因此,蠕虫要逃匿系统的检测必须保证发出的服务请求都属于主机的自体集,蠕虫难以在这样的条件下进行广泛传播.而且,通过防火墙等安全技术的使用,可以在很大程度上减少训练阶段出现的异常网络服务请求.

由于训练时间有限,系统无法保证收集到主机可能发出的所有正常网络服务请求,即难以保证自体字符串集合的完备.因此,一个服务请求在主机的自体字符串集合中没有对应项,并不能确定它是一个恶意的网络服务请求.

为了减少系统的误报,提高蠕虫检测的准确性,需要依据蠕虫的传播特征对发现的可疑网络服务请求进行关联分析,判断其是否由蠕虫引起.系统中进一步的关联分析是以二叉树的形式来进行的.

1.4 利用二叉树结构进行关联分析

检测系统中,探测点通过 Bloom filter 形式的自体字符串集合发现主机的可疑网络服务请求.此外,探测点还收集标识无效服务请求的反向数据包信息,如 ICMP-T3 主机不可达消息、TCP_RST 消息,以发现网络中的无效服务请求.自体字符串集合能够发现可疑的网络服务请求,但不能确定其是否有效.对于未使用的 IP 地址或者主机未启用服务的访问,更可能是蠕虫行为,需要特别加以重视.

网络中的探测点将收集到的可疑信息提交到蠕虫检测系统的控制中心,控制中心基于蠕虫的传播特性对接收的信息进行分析处理.为了对发现的异常信息加以关联,控制中心采用了二叉树的数据结构.二叉树结构简单、处理算法高效,而且二叉树结构能够反映蠕虫传播过程中主机间的联系.

如果蠕虫通过主机甲向主机乙实施感染,在此过程中,主机甲需要向主机乙发出网络服务请求.在蠕虫检测系统中,以子关联关系将网络服务请求的源主机与目的主机联系在一起,目的主机被称为源主机的子关联主机.子关联关系具有传递性,如果主机乙进一步向主机丙发出服务请求,则认为主机丙是主机乙的子关联主机,同时也是主机甲的子关联主机.

将一台主机发出的可疑网络服务请求和该主机的所有子关联主机发出的可疑网络服务请求集中到一个集合,这个集合所涉及的服务被称为主机的关联服务.建立关联服务的集合有利于发现主机及其子关联主机所请求服务的相似性,对于检测蠕虫,特别是检测多维蠕虫具有重要作用.

在蠕虫检测系统中,二叉树将涵盖主机及其子关联主机的所有可疑服务请求.二叉树的每个树节点代表一台主机.二叉树的根节点代表加入树中的第一条服务请求的源主机.树中其他节点分别对应于某条服务请求的目的主机,同时包括了请求的其他信息,如目的端口、协议、时间戳等内容.

蠕虫在传播时常常出现交叉传播的现象,即蠕虫主机向其他已感染的主机发出探测数据包.因此,一台主机在二叉树中可能有多个节点与之对应,检测系统通过节点的时间戳来区分这些节点.

要将一条新的可疑服务请求加入二叉树,首先,在所有现有的二叉树中寻找与服务请求的源主机相对应的节点.如果没有节点对应于该主机,则将服务请求的源主机作为一棵新二叉树的根节点,将服务请求的目的主机作为根节点的左儿子加入树中.如果服务请求的源主机在一棵二叉树中存在对应节点,则选取其中时间戳最新的节点,若该节点没有左儿子,则将服务请求的目的主机作为节点的左儿子加入二叉树;若该节点已有左儿子,则从节点的左儿子出发,递归地访问节点的右儿子,直到找到一个右儿子为空的节点,将服务请求的目的主机作为这个节点的右儿子加入二叉树.采用这种方法构造二叉树,树中的任意一个节点,其左子树中所有节点代表的主机都是它的子关联主机,遍历节点的左子树可以得到节点的关联服务集合.

图 2 所示的二叉树是在如下可疑网络服务请求的基础上建立的:IP 地址为 10.65.1.9 的主机在 $t1$ 时间向 IP 地址为 10.65.1.113 的主机请求(1434 端口,UDP 协议)服务;IP 地址为 10.65.1.9 的主机在 $t2$ 时间向 IP 地址为 10.65.1.20 的主机请求(80 端口,TCP 协议)服务;IP 地址为 10.65.1.113 的主机在 $t3$ 时间向 IP 地址为 202.20.1.5 的主机请求(80 端口,TCP 协议)服务;IP 地址为 10.65.1.9 的主机在 $t4$ 时间向 IP 地址为 10.65.1.15 的主机请求(1434 端口,UDP 协议)服务.

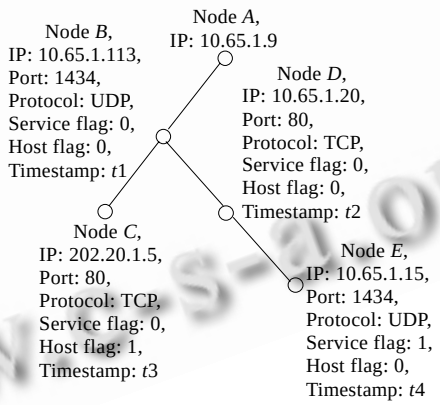


Fig.2 Correlation analysis of suspicious service requests in the form of binary tree

图 2 以二叉树的形式对可疑服务请求进行关联分析

服务标识位和主机标识位用于标识服务请求的有效性.服务标识位用于标识目的主机是否开放了所请求的服务,若服务存在,则标识位为 0,反之为 1.主机标识位用于标识目的主机是否存在,若目的主机存在,则标识位为 0,反之为 1.

在图 2 中,节点 A 标识主机 10.65.1.9,节点 A 的左子树中所有节点代表的主机都是主机 10.65.1.9 的子关联主机,主机 10.65.1.9 的关联服务集合可以通过遍历节点 A 的左子树获得,为{(1434 端口,UDP 协议),(80 端口,TCP 协议)}.

为了定量地分析服务请求的异常,二叉树的每个节点有一个异常值,二叉树中所有节点的异常值之和被称为二叉树的异常值.节点异常值的设置首先考虑服务标识位和主机标识位,如果目的主机没有开放相应服务,或者目的主机不存在,则将赋予较高的异常值.另外,如果所请求的服务隶属于二叉树根节点的关联服务集合,即

二叉树中已出现过类似的网络服务请求,则相应节点将被赋予较高的异常值,因为服务请求的相似性是蠕虫传播的重要特征.二叉树的异常值被作为蠕虫判定的基础.

网络蠕虫通常积极扩散,如果网络内发生了蠕虫事件,相关联的异常服务请求将在短时间内大量出现.因此,蠕虫检测系统对二叉树的生命期进行了限制,着眼于有限时间窗口内服务请求信息的关联.在新节点加入二叉树时,依据树节点的时间戳判断二叉树是否超出生命期,根据需要重新构造二叉树.

1.5 蠕虫攻击的判定策略

实际网络环境具有一定程度的复杂性,当没有蠕虫传播时,网络上也会有一些异常的服务请求和失败的网络连接,控制中心相应地会根据收集到的信息构造二叉树.检测系统需要在没有蠕虫传播的网络环境中进行训练,并以此阶段二叉树的构造情况作为蠕虫判定的基础.

无参 CUSUM 算法[11]被用于监视二叉树异常值的变化.该算法将取样信息作为序列进行分析,全面考虑样本随时间的变化情况.样本短时间的波动不足以触发异常告警,只有样本持续性地偏离限定的取值范围才会导致系统进行告警响应.

检测系统在自体字符串集合建立以后,根据发现的可疑服务请求构造二叉树.在每个时间点,选取异常值最高的二叉树,以 $S_{\max}$ 表示该二叉树的异常值.从利用二叉树进行关联分析的角度来看,对于正常的网络通信,二叉树的异常值通常较低,并且在大部分时间 $S_{\max}$ 值为 0;而网络中出现蠕虫以后,系统依据可疑的网络服务请求所构建的二叉树在短时间内通常会迅速增长,二叉树的异常值将迅速提高.

应用无参 CUSUM 算法对 $S_{\max}$ 进行监视.采样的时间点以参数 n 表示,$X_n(n=1,2,3,\ldots)$表示在各采样点参数 $S_{\max}$ 的值,以 $E(X_n)$表示 X_n 的均值.正常情况下,$X_n(n=1,2,3,\ldots)$是一个均值接近于 0 的序列,采用$\mu_{\max}$表示 X_n 在正常网络通信情况下的均值.当网络中出现蠕虫以后,X_n 的值将明显增大.由于二叉树的生命期被加以限定,二叉树不会无限制地增长.二叉树异常值的累积和 X_n 在蠕虫攻击发生后会在一个定值附近波动,以 $E(X_n)=\mu_{\max}+h$ 表示蠕虫攻击出现后 X_n 的均值.

无参CUSUM算法对其研究对象$Z_n(n=1,2,3,\ldots)$存在限制条件:假设变化在时间点m发生,对于$Z_n(1\le n<m)$,Z_n的均值应当为负数;在变化发生以后,即对于 $Z_n(n\ge m)$,Z_n 的均值应当为正数.为了使蠕虫检测系统的监控对象满足无参 CUSUM 算法的条件,令

$$Z_n=X_n-\beta \tag{1}$$

在公式(1)中,参数β是一个定值,它的取值范围是$\mu_{\max}<\beta<\mu_{\max}+h$.通过进行这样的处理,$Z_n$ 可以满足无参 CUSUM 算法的限制要求.

采用递归形式的无参 CUSUM 算法对 Z_n 进行描述:

$$\begin{cases} y_0 = 0 \\ y_n = f(y_{n-1} + Z_n) \end{cases} \tag{2}$$

其中,函数 $f(x)=\begin{cases} x, & x>0 \\ 0, & x\le 0 \end{cases}$.根据无参 CUSUM 算法的限制条件,当 $n<m$ 时,即变化发生以前,Z_n 的均值 $E(Z_n)=\mu_{\max}-\beta<0$,$Z_n$ 的序列围绕$\mu_{\max}-\beta$波动.y_n 值的增长要求 Z_n 持续为正值,当 $n<m$ 时,Z_n 短暂性地处于正值并不会使 y_n 值迅速提升.而当 $n\ge m$ 时,Z_n 的均值 $E(Z_n)=\mu_{\max}+h-\beta>0$,$Z_n$ 的序列围绕$\mu_{\max}+h-\beta$波动.由于此时 Z_n 在大部分取样时间处于正值,因此 y_n 值将迅速增长.对 y_n 值的变化进行监视,可以及时发现蠕虫攻击.引入函数 $d_N(y_n)$判断网络中是否出现了蠕虫攻击:

$$d_N(y_n)=\begin{cases} 0, & y_n < N \\ 1, & y_n \ge N \end{cases} \tag{3}$$

公式(3)中,N 是判定阈值.当 y_n 值持续增长并超过阈值 N 时,则判定网络中出现了蠕虫攻击.在 y_n 值小于阈值 N 时,认为网络运作正常.

图 3 给出了测试实验中 $E(X_n)$值的变化情况.实验中,二叉树的生命期限定为 30s,实验蠕虫基于 TCP 协议传播,并行连接数量为 5.实验蠕虫的扫描策略与 Code Red II 相同,采用本地地址优先的扫描策略.蠕虫被限定在

600s 时开始在网络传播.在蠕虫攻击出现之前,X_n 的均值 $E(X_n)$接近于 0;在 600s 之后,$E(X_n)$明显提高.

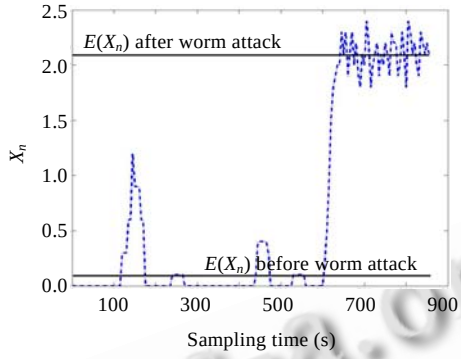


Fig.3 Comparison of $E(X_n)$ before and after worm attack

图 3 蠕虫攻击前、后 $E(X_n)$值的对比

为了满足无参 CUSUM 的限制要求,需要将 X_n 转化为 Z_n 进行处理.X_n 与 Z_n 之间的关系为 $Z_n=X_n-\beta$.在转化过程中涉及到参数β的选择,参数β的选择决定了系统的灵敏度.如果β值过小,对于正常的网络通信,Z_n 也常常为正值,容易导致 y_n 值迅速增长,造成误报;如果β值过大,蠕虫的传播行为甚至不能使 Z_n 达到正值,y_n 值相应地不会明显增长,从而造成系统漏报.

采用实验的方法对β的取值进行选择.实验在自体字符串集合已建立的基础上进行,进一步收集了 24 小时的不包含蠕虫传播行为的正常网络通信数据,以二叉树结构进行关联分析,二叉树的生命期限定为 30s.实验中,$S_{\max}$ 的样本平均值以$\mu_{\max}$ 表示,样本均方差以$\sigma_{\max}$ 表示.选择不同的β值,网络中的 $S_{\max}$ 超过β作为一次告警,表 1 为实验结果.一个可疑的网络操作往往会支配一段连续时间内的 $S_{\max}$ 值,表 1 中的统计结果将这样的可疑操作记为一次告警.

X_n 是对 $S_{\max}$ 的取值.引起告警意味着 $X_n-\beta>0$,即 Z_n 为正值.在无参 CUSUM 算法中,Z_n 为正值,可以视为被系统作为异常接受进一步的处理,可能促使 y_n 值的增长.理想情况下,对于正常的网络通信,Z_n 在大部分时间应当为负值.

Table 1 Results of choosing different value for β

表 1 选择不同β值的测试结果

β	Alarms/minute
$\mu_{\max}$	0.140
$\mu_{\max}+0.5\sigma_{\max}$	0.104
$\mu_{\max}+\sigma_{\max}$	0.079
$\mu_{\max}+1.5\sigma_{\max}$	0.052

参数β的选择是蠕虫检测系统的检测率和误报率的折衷.网络蠕虫的扫描策略和传播速度的差异将影响到蠕虫攻击发生以后 X_n 的取值范围,因此无法事先推断在蠕虫攻击以后均值增量 h 的具体值.按照表 1,当$\beta=\mu_{\max}+0.5\sigma_{\max}$ 时,系统中的告警已经较少,或者说在正常的网络环境中,X_n 超过$\mu_{\max}+0.5\sigma_{\max}$ 是小概率事件.将参数β代入 Z_n 中,在变化发生之前,$E(Z_n)=-0.5\sigma_{\max}<0$;在变化发生以后,$E(Z_n)=h-0.5\sigma_{\max}$.有可能有传播非常隐秘、非常缓慢的蠕虫,在系统中发起攻击以后,X_n 均值的增量 $h<0.5\sigma_{\max}$,由于攻击没有使 Z_n 到达正值,异常不会被发现.考虑到无论如何降低β的值,都有可能有传播更为隐秘的蠕虫逃匿检测,而且所设计的蠕虫检测系统主要以中、高速传播的蠕虫作为检测对象,因此,将参数β设置为$\mu_{\max}+0.5\sigma_{\max}$.图 4 是采用该$\beta$参数所得到的 Z_n 随时间的变化情况.

在确定 Z_n 以后,使用无参 CUSUM 算法对 Z_n 进行处理,将 Z_n 转化为 y_n.y_n 是对 Z_n 变化的累加,只有在 Z_n 持续为正值时,y_n 值才会迅速提高.图 5 给出监测到的 y_n 值随时间的变化情况,从图中可以看出,在蠕虫攻击发生以

后,y_n 值迅猛提高;而在蠕虫攻击发生以前,Z_n 短暂性地增长对 y_n 值的影响很小.

无参 CUSUM 算法还需要确定告警阈值 N,通过该阈值判断网络中是否出现了蠕虫攻击.在系统中通过充分训练,获得 y_n 在训练阶段对于正常网络通信达到的最大值,作为告警阈值 N,如图 5 所示.阈值 N 的选取是误报率和检测时效性之间的折衷:如果阈值选择得较低,正常的网络波动也可能使 y_n 超过设定的阈值,造成误报;如果阈值 N 选择得较高,则可以减少误报的出现,但发现蠕虫的时间将在一定程度上有所延长.

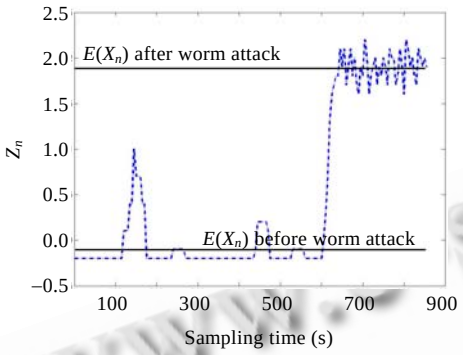


Fig.4 Comparison of $E(Z_n)$ before and after worm attack

图 4 蠕虫攻击前、后 $E(Z_n)$值的对比

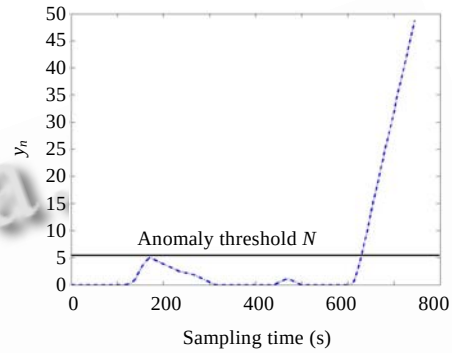


Fig.5 y_n increases dramatically after worm attack

图 5 y_n 值在蠕虫攻击发生后出现明显增长

2 测试实验

GTNetS(Georgia Tech Network Simulation)[12]是一个功能全面的网络仿真平台,利用该平台对蠕虫检测系统进行了测试.评价蠕虫检测机制主要考虑到两个标准:第 1 个标准检测机制是针对蠕虫传播的响应速度和抑制能力,表现为网络中蠕虫感染主机的数量变化;第 2 个标准检测机制是针对正常网络通信的影响,这种影响应当是越小越好.实验中主要考虑网络中应用检测机制以后,网络通信的响应时间是否显著延长.Web 服务在网络上应用最为广泛,具有普适性.实验中以 Web 服务的请求和应答作为正常网络通信行为的代表进行研究,着重考察在应用蠕虫检测系统的网络中,Web 客户端从发出服务请求到接收到服务器的响应,其间隔时间的长度简称为 Web 服务的响应时间.

所提出的基于阳性选择的蠕虫检测系统着眼于蠕虫的检测,可以与各种蠕虫抑制措施结合使用.在测试实验中,检测系统一旦做出蠕虫告警,将对蠕虫所攻击的网络服务进行限制,不允许感染蠕虫的主机发出针对相应服务的访问请求,避免蠕虫进一步扩散.

测试实验在一个中型网络上进行,拓扑结构如图 6 所示.网络中包括 6 个子网,其中 5 个子网中主机数量为 20,一个子网中主机数量为 10.每个子网在图中呈现为一个星形结构,子网边缘位置的节点为普通的网络主机,位于中央的节点为交换机.网络主机与交换机之间的链路带宽统一为 10M,链路延时统一为 10ms.此外,3 个路由器节点负责将几个子网连接在一起,路由器与交换机之间的链路带宽统一为 100M,链路延时统一为 10ms.路由器之间的链路带宽为 500M,链路延时为 10ms.

Code Red 蠕虫的扫描率约为 6 次/秒[13],Slammer 蠕虫的扫描率约为 4 000 次/秒[14],两种蠕虫是非常典型的中、高速传播的蠕虫.参照这两种蠕虫,测试实验采用了扫描率为 6 次/秒和 4 000 次/秒的 UDP 蠕虫.两种测试蠕虫的负载大

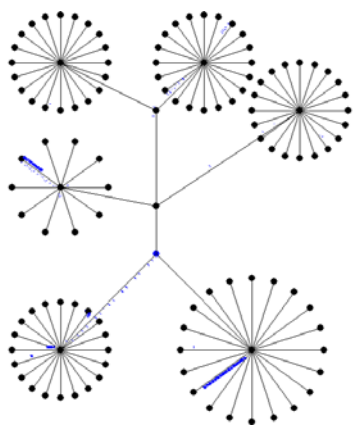


Fig.6 Network topology used in the experiment

图 6 测试实验使用的网络拓扑

小统一为 500 字节,实验利用 GTNetS 平台的随机扫描策略传播.实验时间的总长度为 100s.蠕虫通过网络中的一台主机发布,设定在 50s 时开始扩散.网络中易感主机的比例为 60%.

为了获取 Web 服务响应时间的数据,实验中一些网络节点被设置为 Web 服务器,一些网络节点被设置为 Web 客户机.每个 Web 客户机随机地选择时间以及 Web 服务器进行访问,并记录每次访问的响应时间.实验结束后,以分布函数的形式对 Web 服务响应时间的情况进行统计分析.

用 X 表示 Web 服务的响应时间,它是一个随机变量.函数 $F(x)=P(X\le x)$被称为 X 的分布函数,其中 x 代表一个实数.函数 $F(x)$的含义是响应时间 X 的值小于 x 的概率.响应时间以秒为单位,如果 x 为 0.5,则 $F(0.5)$表示 Web 服务的响应时间小于 0.5s 的概率.

网络通信的响应时间受两类因素的影响:一类因素与网络拓扑相关,主要考虑链路带宽以及链路延时等因素的影响,在网络结构不变的情况下,此类因素的影响是固定的;另一类因素具有动态性,测试实验主要涉及两方面:首先是网络的拥塞程度,在网络较为拥塞的情况下,数据包在网络设备上排队等待处理的时间较长.当网络上有蠕虫传播时,如果蠕虫传播较快,很容易由于发送的数据包过多造成网络拥塞,延长通信的响应时间;其次,要考虑蠕虫检测机制对数据包传输的影响,蠕虫检测一般涉及对数据包的分析、处理,这个过程将增加通信的响应时间.在网络带宽、链路延时等信息固定的情况下,通信的响应时间越短越理想.以分布函数的形式分析通信的响应时间,分布函数的值越高越理想.

实验中选择了 Counter Malice[15],DAW[16],Packet Matching[17]以及 Virus Throttle[18]这 4 种蠕虫防护方法与所提出的基于阳性选择的蠕虫检测方法进行对比.

图 7 和图 8 分别为扫描率为 6 次/秒的 UDP 蠕虫在网络中传播,采用 Counter Malice,DAW,Packet Matching, Virus Throttle 以及基于阳性选择的蠕虫检测方法,网络中蠕虫感染主机的数量增长情况以及 Web 通信响应时间的分布函数的情况.图 9 和图 10 是扫描率为 4 000 次/秒的 UDP 蠕虫的实验结果.

从扫描率不同的两种蠕虫的实验结果可以看出,基于阳性选择的蠕虫检测方法对于高速传播的蠕虫以及传播较为缓慢的蠕虫同样有效.与其他几种蠕虫防护方法相比,基于阳性选择的蠕虫检测方法及时发现了网络中的蠕虫并加以抑制,网络中感染蠕虫的主机数量最少.此外,在两组实验中,用 x 代表 Web 通信响应时间,采用基于阳性选择的蠕虫检测方法,响应时间的分布函数的值 $F(x)$最高.这意味着采用所提出的蠕虫检测方法对网络进行防护,从整体上看,Web 通信的响应时间最短;或者说,这种蠕虫检测方法在限制蠕虫传播的同时对于正常网络通信的影响最小.

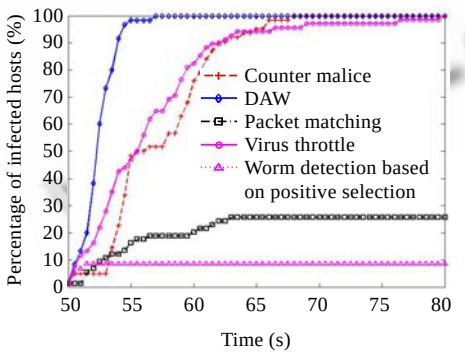


Fig.7 Hosts infected by UDP worm with scan rate of 6/s

图 7 扫描率为 6 次/秒的 UDP 蠕虫的主机感染情况

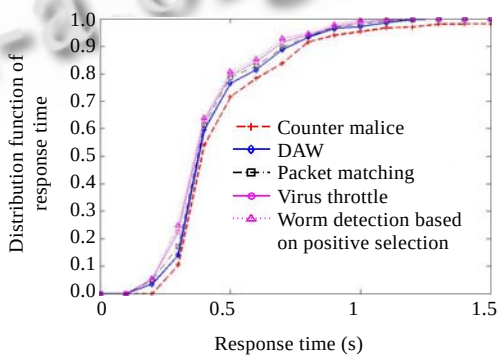


Fig.8 Distribution functions of Web response time using UDP worm with scan rate of 6/s

图 8 扫描率为 6 次/秒的 UDP 蠕虫传播时 Web 通信响应时间的分布函数

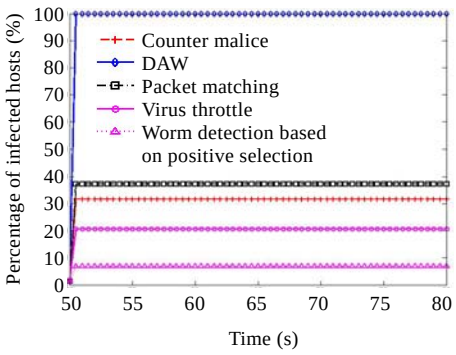


Fig.9 Hosts infected by UDP worm with scan rate of 4000/s

图 9 扫描率为 4000 次/秒的 UDP 蠕虫的主机感染情况

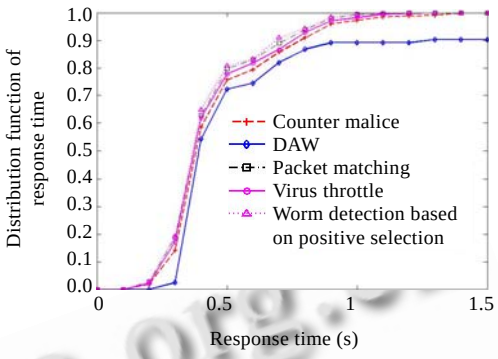


Fig.10 Distribution functions of Web response time using UDP worm with scan rate of 4000/s

图 10 扫描率为 4000 次/秒的 UDP 蠕虫传播时 Web 通信响应时间的分布函数

3 结 论

论文针对蠕虫的传播特性,以阳性选择算法为基础设计了一种蠕虫检测系统.该检测系统以自体字符串的形式表示主机的正常网络服务请求,通过构造 Bloom filter 形式的自体字符串集合发现主机发出的可疑网络服务请求.根据蠕虫的网络传播特点,系统以二叉树的形式对发现的可疑信息进行关联分析,采用无参 CUSUM 算法监视二叉树异常值的变化,从而及时发现蠕虫传播.在 GTNetS 仿真平台上对基于阳性选择的蠕虫检测方法进行了仿真,将它与 Counter Malice,DAW,Packet Matching,Virus Throttle 这 4 种蠕虫防护方法进行比较.实验结果表明,采用基于阳性选择的蠕虫检测系统,由于蠕虫被及时发现并受到控制,网络中被蠕虫感染的主机数量最少.同时,这种方法在实施蠕虫检测的同时对正常网络通信的影响最小.

References:

[1] Wen WP, Qing SH, Jiang JC, Wang YJ. Research and development of Internet worms. Journal of Software, 2004,15(8):1208−1219 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1208.htm

[2] Qing SH, Wang C, He JB, Li DZ. Research and development of instant messaging worms. Journal of Software, 2006,17(10): 2118−2130 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2118.htm [doi: 10.1360/jos172118]

[3] Zheng H. Internet worm research [Ph.D. Thesis]. Tianjin: College of Information Technologies Science, Nankai University, 2003. (in Chinese with English abstract).

[4] Forrest S, Perelson AS, Allen L, Cherukuri R. Self-Nonself discrimination in a computer. In: Rushby J, Meadows C, eds. Proc. of the IEEE Symp. on Research in Security and Privacy. Los Alamitos: IEEE Computer Society Press, 1994. 202−212.

[5] D'haeseleer P, Forrest S, Helman P. An immunological approach to change detection: algorithms, analysis and implications. In: Proc. of the IEEE Symp. on Security and Privacy. Oakland: IEEE Computer Society Press, 1996. 110−119. http://www.cs.unm.edu/~forrest/publications/ieee-sp-96-neg-selec.pdf

[6] Stibor T, Bayarou KM, Eckert C. An investigation of *r*-chunk detector generation on higher alphabets. In: Deb K, ed. Proc. of the Genetic and Evolutionary Computation Conf. Seattle: Springer-Verlag, 2004. 299−307.

[7] González FA, Dasgupta D. Anomaly detection using real-valued negative selection. Genetic Programming and Evolvable Machines, 2003,23(3):383−403.

[8] Ji Z, Dasgupta D. Augmented negative selection algorithm with variable-coverage detectors. In: Proc. of the 2004 Congress on Evolutionary Computation. IEEE Press, 2004. 1081−1088. http://ais.cs.memphis.edu/files/papers/CEC_1-2004.pdf

[9] Bloom B. Space/Time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970,13(7):422−426. [doi: 10.1145/362686.362692]

[10] Broder A, Mitzenmacher M. Network applications of Bloom filters: A survey. Internet Mathematics, 2004,1(4):485−509.

[11] Brodsky BE, Darkhovsky BS. Nonparametric Methods in Change-Point Problems. Dordrecht: Kluwer Academic Publishers, 1993.

[12] George FR, Monirul IS, Lee WK. Simulating Internet worms. In: Doug D, Peter GH, eds. Proc. of the IEEE Computer Society 12th Annual Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. Washington: IEEE Computer Society, 2004. 268−274.

[13] Zou CC, Gao LX, Gong WB, Towsley D. Monitoring and early warning for Internet worms. In: Jajodia S, ed. Proc. of the 10th ACM Conf. on Computer and Communications Security. New York: ACM, 2003. 190−199.

[14] Moore D, Paxson V, Savage S, Shannon C, Staniford S, Weaver N. Inside the slammer worm. IEEE Security & Privacy, 2003,1(4): 33−39. [doi: 10.1109/MSECP.2003.1219056]

[15] Mohamed A, George FR. Evaluation of worm containment algorithms and their effect on legitimate traffic. In: Cole J, ed. Proc. of the 3rd IEEE Int'l Workshop on Information Assurance. Washington: IEEE Computer Society, 2005. 33−42.

[16] Chen SG, Tang Y. Slowing down Internet worms. In: Proc. of the 24th Int'l Conf. on Distributed Computing Systems. Washington: IEEE Computer Society, 2004. 312−319. http://reference.kfupm.edu.sa/content/s/l/slowing_down_internet_worms__1114146.pdf

[17] Chen X, Heidemann J. Detecting early worm propagation through packet matching. Technical Report, ISITR-2004-585, University of Southern California, 2004. 1−13.

[18] Jamie T, Matthew MW. Implementing and testing a virus throttle. In: Paxson V, ed. Proc. of the 12th USENIX Security Symp. Washington: USENIX Association, 2003. 285−294.

附中文参考文献:

[1] 文伟平,卿斯汉,蒋建春,王业君.网络蠕虫研究与进展.软件学报,2004,15(8):1208−1219. http://www.jos.org.cn/1000-9825/15/1208.htm

[2] 卿斯汉,王超,何建波,李大治.即时通信蠕虫研究与发展.软件学报,2006,17(10):2118−2130. http://www.jos.org.cn/1000-9825/17/2118.htm [doi: 10.1360/jos172118]

[3] 郑辉.Internet 蠕虫研究[博士学位论文].天津:南开大学信息技术科学学院,2003.

洪征(1979－),男,江西南昌人,博士,讲师,主要研究领域为网络安全,人工智能.

吴礼发(1970－),男,博士,教授,博士生导师,主要研究领域为网络安全,软件可靠性.