基于流量特性的僵尸网络检测方法研究

张阳[1, 2]，程光[1, 2]

（1.东南大学计算机科学与工程学院，南京，211189；2.计算机网络与信息集成教育部重点实验室，南京，211189）

摘　要：僵尸网络是指通过多种手段在多台计算机中植入恶意程序，使僵尸控制者能相对方便和集中的控制这些计算机，向这些受控制的计算机发布各种指令进行相应恶意活动的攻击网络;现有的基于流量行为的僵尸网络检测技术往往在特征提取上没有针对性，只是针对单个报文或单个流，并在特征聚类上采用多维聚类，聚类效果差，导致整体的检测效果低下；本文通过研究僵尸客户端与 C&C 服务器交互时产生的流量特性，以流序列为研究对象，提取流间隔时间、流持续时间、流字节数、交互频率等特征，采用一维聚类及与之对应的评分机制，与未知流量进行匹配，从而达到检测僵尸网络的目的，并取得了良好的效果；并在此基础上，实现了一个小型的僵尸网络检测系统。

关键词：僵尸网络；流量特性；频率提取；机器学习

Botnet Detection by Analyzing Behaviors in Network Traffic

Zhang Yang[1, 2], Cheng Guang[1, 2]

(1. School of Computer Science & Engineer, Southeast University, Nanjing, 211189;

2. Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing, 211189)

Abstract: A botnet is a kind of malicious network which implants malicious programs into many computers through a variety of means, so that the controller can control these computers in a convenient and centralized way; the controller can do many attacks on the Internet by using all of these computers. The existing botnet detection technology based on flow behavior often focus on a single packet or a single stream, and use multidimensional clustering which will lead to the detection of low effect. This article puts forward a botnet detection method by analyzing behaviors in network traffic generated by botnet client and C&C server, and extract features like time interval of flows, duration of connections, number of bytes in flows and period of flow sequence to detect new traffic; In addition, this article use one-dimensional clustering and a botnet detecting system based on this method is established.

Key words: Botnet; Traffic behaviors; Frequncy extracting; Machine learning

1 引言

僵尸网络是指通过各种手段在多台计算机中植入恶意程序，使僵尸控制者能相对方便和集中的控制这些计算机，向这些受控制的计算机发布各种指令进行相应恶意活动的攻击网络；按照控制信道使用互联网协议的不同，可分为 IRC 僵尸网络、HTTP 僵尸网络、P2P 僵尸网络[1]。僵尸网络控制者通过 C&C 服务器（命令与控制服务器）向僵尸客户端发送控制指令，进行 DDOS、发送垃圾邮件、点击欺诈等恶意行为。

自从 1999 年 3 月第一个恶意 IRC 僵尸程序“pretty park”被发现以来[2]，僵尸网络迅速发展，对互联网造成了巨大危害，僵尸程序及节点结构越来越复杂，从最初简单的 IRC bot，发展成为集自我复制、漏洞扫描、自动传播、信息窃取、反侦测等诸多功能的僵尸网络系统。

僵尸网络检测方法主要分为两种，第一种基于分析可疑二进制可执行文件，这种方法一般是通过蜜罐捕获可疑二进制文件，分析攻击记录和活动记录，发现并摧毁僵尸网络，这种方法可靠性好，但是分析周期长，消耗资源大，实时性差；第二种基

作者简介：张阳，（1989-），男，硕士研究生，E-mail: zhang_yang0108@hotmail.com；程光，（1973-），男，教授，博导 E-mail: gcheng@njnet.edu.cn.

于分析网络流，通过 DPI（深度包检测）方法分析网络报文的字段特征或者是分析流量的行为特征与关联特征，从而找出僵尸流量与正常流量的差异性，以此发现僵尸网络，这种方法实时性好，消耗资源小，部署简单，但是由于网络流量的复杂性，准确性相对较低。

本文提出的方法属于第二种，通过分析僵尸客户端与 C&C 服务器通信过程中产生流量的行为特性，提取行为特征，以此区分未知流量，发现僵尸网络，并具有较好的检测效果。相比于以往的基于流量行为的僵尸网络检测技术，本文主要贡献如下：

1 现有的基于流量行为的僵尸网络检测技术往往在特征提取上没有针对性，只是针对单个报文或单个流，这并不足以描述僵尸客户端在与 C&C 服务器通信时产生的流量特性，本文以单向流的流序列为研究对象，即以客户端与服务器的一次完整通信过程为单位，提取流间隔时间、流持续时间、流字节数等基本特征以及交互频率等整体特征，更具有针对性。

2 本文采用一维聚类及相应的评分机制建立僵尸特征模型和对未知流量的检测，而不是直接采用通常的多维聚类方法，因为即使采用了各种降维技术，多维聚类往往复杂度过高，且聚类结果较差，达不到应有的检测效果。

3 本文在提取流序列基本特征的同时，更针对性的提取流序列的整体周期特征，因为周期性是僵尸网络通信中一个极为重要的特征，本文使用能量谱与循环自相关函数同时描述周期性，相当于加大了周期性在评分机制中的权重，并取得了良好效果。

本文在接下来的几章中，详细描述了算法与系统的实现。第二章叙述了国内外今年对于僵尸网络的研究现状；第三章叙述了算法的具体实现，即特征的选择和提取过程及聚类与分类方法；第四章叙述了系统的实现与实验方法及实验结果；最后第五章总结了本文描述的方法，并指出了存在的不足和可改进之处。

2 相关工作

2005-2006 年，僵尸网络检测技术开始得到关注，针对不同的僵尸网络类型，开始出现各种基于网络流量的僵尸网络检测技术。总体来说，可分为需要使用 DPI 和不需要使用 DPI 两种。

需要使用 DPI 技术的方法一般关注报文的具体载荷内容，包括 IRC 昵称、IRC 命令序列、报文特征二进制片段、应用层 HTTP 协议内容、应用层 DNS 协议内容、应用层 SMTP 协议内容等。

Rishi 利用已知的僵尸昵称模式来对新的 IRC 昵称进行评分，若新的昵称结构与已有的僵尸昵称结构相似，则判定为僵尸昵称[3]。

王威、方滨兴、崔翔等人关注同一 IRC 频道下昵称的相似性，从而不需要先验知识；同时他们也观测僵尸程序登录僵尸频道后产生的 IRC 命令序列，通过检测命令序列的相似性进行判断[4]。

Konrad Rieck 等人记录恶意代码产生的网络流量，提取出特征载荷，并与正常的流量提取出的特征载荷比较，剔除无效特征，最后用这些特征载荷去分析新的流量，并进行识别[5]。

Jae-Seo Lee 等人通过分析 HTTP 僵尸程序“BlackEnergy”，发现其周期性访问两个 C&C 服务器，一个是使用者设定的服务器，一个是僵尸程序编写者设定的服务器，通计算一个 IP 访问 web 服务器记录的自由度和标准差来表征周期性程度[6]。

Hyunsang Choi 等人提出了 BotGAD 系统，首先把 DNS 流量按照 IP 进行分类，并以 IP 为行，时间片为列形成矩阵，如果某一时间片内该 IP 有 DNS 请求，即将对应位置 1，否则置 0，然后计算向量组的平均相似度，集中程度，周期度，设定相应的阈值用以判别[7]。

Ian Castle 等人使用了一种名为“Plato”算法的垃圾邮件检测算法，首先，通过分析 SMTP 邮件首部格式和内容，构造邮件模板，然后，对每个邮件产生的模板进行 MD5 的 hash 计算，同一个僵尸网络的垃圾邮件模板应该被 hash 到一个值，由此来进行僵尸网络的判定[8]。

不需要使用 DPI 技术的方法一般关注网络流量的宏观特性，包括对单报文和“流”（两个端系统之间的一次特定的会话过程）的分析，以及字节数、字节速率、包速率、时间间隔、持续时间等特征的提取；这类方法通过对僵尸流量周期性和群体相似性的研究，发掘僵尸流量与正常流量在周期与群体性上的差异性，以此发现僵尸网络。

Binbin Wang 等人采用请求字节数、响应字节数、有效净报文（除去重传、副本、空载荷等）来描述一次交互过程，采用 X-means 聚类，并利用循环自相关方法探测聚类中连接的周期性行为[9]。

BotSniffer 是一个针对僵尸 IP 群体性行为的检测系统，该系统通过 RCDC 算法（检测一个响应集群产生突发可疑响应高峰，判断这个集群属于某个僵尸网络的概率）和 RCHC 算法（一个响应集群产生响应的相似度来判断这个集群属于僵尸网络的概率）来对僵尸网络进行检测[10]。

BotMiner 是一个比 BotSniffer 更普遍的基于群体行为的僵尸网络检测系统，它提出了 C-flow 的概念，并通过判断 C-flow 的相似性来进行僵尸网络检测[11]。

特别的，Florian Tegeler 提出了 Trace 的概念，将多个流关联起来，从而更好的表征两个 IP 之间的交互过程，研发的 BotFinder 系统通过对各类僵尸程序产生的 Trace 特征的提取，在对未知流量的检测中获得了极佳的检测效果[12]。本文借鉴了 BotFinder 系统中 Trace 的概念。

Guofei Gu 的 BotHunter 是一个将两类检测技术结合的僵尸网络检测系统，采用 SLADE 子系统进行 DPI 分析，SCADE 进行流量行为分析，建立僵尸网络感染会话过程，并进行判断[13]。

3 算法分析与实现

3.1 报文信息封装

3.1.1 流

流：在一段时间内，一个源地址和目的地址之间传输单向报文流，所有报文具有相同的传输层源、目的端口号、协议号和源、目的地址，即五元组内容相同[14]。

流描述的是两个端系统之间一次特定的通信活动。将报文组成流的意义在于将一次通信行为作为一个整体进行研究，而忽略单个报文之间的差异性，关注的是这次通信活动整体的特性，如流持续时间、流字节数、流结束原因等。

为更精确的描述僵尸主机与 C&C 服务器通信过程中产生流量的特性，本文使用的是单向流（如端系统 A 向端系统 B 进行 TCP 连接，A 向 B 先发送 SYN 报文，B 反馈 SYN-ACK，A 再反馈 ACK 报文，之后进行正常通信，最后以 FIN、RST 或超时结束这次通信活动，那么在这次通信活动中，A 向 B 发送的所有报文序列为一个 A 到 B 的单向流，AB 之间所有的报文序列为一个双向流）。

3.1.2 Trace

Trace（本文亦称为“流序列”）：在一段时间内，一个特定源地址与一个特定目的地址及特定目的端口之间通信所产生的流的序列[12]。

流序列描述的是一段时间内两个端系统之间同种通信活动的整体特性。如端系统 A（sIP）请求端系统 B（dIP）的 80（dPort）端口，一段时间内连续请求多次，AB 产生多个流，则这些流按流开始时间排序组成的序列为一个流序列。

在一个流序列中，既有 sIP 到 dIP 的 dPort 的单向流，亦有 dIP 的 dPort 到 sIP 的相反方向的单向流，通过研究流序列中多个流之间的整体特性，如平均流时间间隔、平均流持续时间、平均流字节数等，从而更好的反应 sIP 与 dIP 的通信行为。

3.2 流序列特征

本文采用单向流组成的流序列，相比于双向流，单向流能更精细的描述一次通信活动。单向流与双向流的差异在于，单向流没有规定源宿的方向，对于一组未知流量来说，规定方向组成的双向流很容易造成信息的丢失，因为本文研究对象是（sIP，dIP，dPort）形式的流序列，对于（sIP，dIP，sPort，dPort，protocol）形式的双向流，源宿互换可以产生完全不同的流序列，如果 C&C 服务器正好在 dIP 中，则结果正常，如果在 sIP 中，僵尸主机则作为 dIP，由于僵尸主机每次与 C&C 服务器通信所使用的端口（dPort）可能是操作系统自动分配的随机端口，原本同一种通信流，将会被分配到不同流序列中，对检测效果产生巨大影响。

为防止此类结果发生，本文在利用单向流拼装流序列时，一个流 A 将和它的反向副本 B 同时进行拼装，即一个流将会以流本身和其反向副本的形式出现在两个流序列中。反向副本 B 定义如下：对于一个单向流 A 以（源 IP，宿 IP，源端口，宿端口，协议）形式有（sIP，dIP，sPort，dPort，protocol），将 A 源宿互换成为 B，以同样形式成为（dIP，sIP，dPort，sPort，protocol），则 B 为 A 的反向副本，值得注意的是，B 并不是一个新流，只是 A 的一个副本，本文亦称为目的流。

本文关注流序列的 3 对基本的特征及 2 个整体特征，共 8 个特征；基本特征有平均源流时间间隔、平均源流持续时间、平均源流字节数、平均目的流

时间间隔、平均目的流持续时间、平均目的流字节数；对于整体，有描述流周期性特征的能量谱密度及循环自相关系数。

3.2.1 流序列基本特征

平均流时间间隔：流序列中两个相邻流之间起始时间的差值的平均。

平均源流持续时间：流序列中所有源流持续时间的平均。

平均源流字节数：流序列中所有源流字节数的平均。

按照参与分析时是流本身或逆向副本的不同，分为平均源流时间间隔与平均目的流时间间隔。

对于僵尸网络，僵尸主机大多数时间处于等待状态，为保持连通性，僵尸主机会定时与相应的服务器发送报文并接受回馈报文确保自身的连通性，而这些通信行为在流时间间隔、流持续时间、流字节数这些基本特征上有很强烈的相似性，而正常流量则是随机的。

3.2.1 流序列周期特征

为描述流序列的周期特性，首先对流序列进行01序列化，序列化规则如下：

$$f(m) = 1 \quad (m = \lfloor (S(n)-S_{min})/t \rfloor ,\ 0 \le n \le N-1) \qquad (1)$$

f(m)的其他项取值为0，其中，S(n)为流序列中所有流起始时间序列，N 为流序列中流个数，S_{min}为起始时间最小的流的起始时间，t 为序列化时间间隔，为使 m 的值不至于太大，并保证一定的精确性，本文取平均流时间间隔的 1/4，即 $t = Interval_{avg}/4$，则产生一个 $0 \le m \le M-1$ 的 01 序列 f。

序列 f 描述了流起始时间的分布情况，本文通过计算 f 的能量谱密度及循环自相关系数来描述 f 的周期性。

f 的离散傅里叶变换：

$$F[k] = \sum_{m=0}^{M-1} e^{-i\frac{2\pi}{M}mk} f[m] \quad k = 0,...,M-1 \qquad (2)$$

f 的能量谱密度函数：

$$\Phi[k] = \frac{F[k]F^{*}[k]}{2\pi} \quad k = 0,...,M-1 \qquad (3)$$

f 的离散自相关函数：

$$R[k] = \sum_{-\infty}^{+\infty} f[m]f[m-k] \quad k = 0,...,M-1 \qquad (4)$$

f 的功率谱密度函数：

$$\Psi[k] = \sum_{m=0}^{M-1} e^{-i\frac{2\pi}{M}mk} R[m] \quad k = 0,...,M-1 \qquad (5)$$

f 的离散循环自相关函数：

$$R'[k] = \sum_{m=0}^{M-1} f[m]f[(m+1)\%M] \qquad (6)$$

Florian Tegeler 等人使用功率谱密度来描述 f 的周期性[12]，即 f 的自相关函数的傅里叶变换，自相关函数的计算复杂度为 $O(n^2)$，傅里叶变换使用 FFT 计算复杂度为 $O(nl_{og}n)$，则总计算复杂度为 $O(n^2)$，本文采用能量谱密度描述 f 的周期性，实验结果表明，能量谱同样能很好的描述序列 f 的周期性，由公式(3)得知，能量谱等于 f 的傅里叶变换 F 与其共轭函数 F^{*}的乘积，利用 FFT，计算复杂度是 $O(nl_{og}n)$，实验结果如图 1 所示，实验使用的数据是 10 个 SDbot 产生的 10 个流序列，生成的 f 序列规模从 32 到 16384 不等。

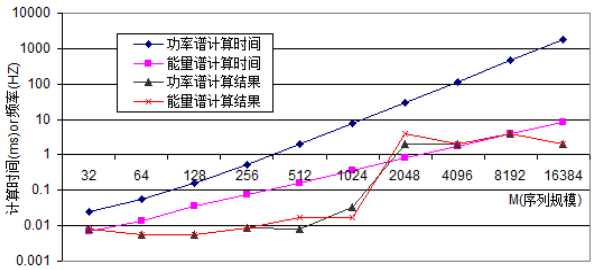

图 1 功率谱与能量谱比较（对数图）

由图 1 可以得知，随着 f 规模的增加，功率谱的计算时间增加极快，而能量谱计算时间则保持在一个较低水平，而他们的计算结果则基本一致。

为计算 FFT，本文选取 f 序列的前 p 个元素构成的子序列，满足：

$2^n = p,\ p \le M,\ 2^{n+1} > M \quad n, p$ 为整数 (7)

在得到能量谱密度函数 Φ(k)后，选取使 Φ 最大的 $k(k \ne 0)$，由公式(8)计算得到频率 freq。

$$freq = k / p / t \qquad t = Interval_{avg}/4 \qquad (8)$$

$$\omega = \max(R'[k]/R'[0]) \quad k = 1,...,M-1 \qquad (9)$$

$acf\_freq = 1/(k_\omega * t)$　　$t = Interval_{avg}/4$ (10)

同样的，对于 f 的离散循环自相关函数，利用公式(9)和(10)计算出频率 acf_freq。

freq 和 acf_freq 描述了流序列的周期性，僵尸主机在大多数时间内都处于等待状态，为保持与 C&C 服务器的连通性，会周期性的与 C&C 服务器交换信息，产生周期性的流量，利用能量谱与循环自相关方法，能将隐含在流序列中的周期性挖掘出来。

3.3 聚类与分类方法

给定一组待训练的僵尸流序列，对于 3.2.1 节所述的 6 个基本特征及 3.2.2 节所述 2 个周期特征，本文采用按特征分别聚类的方法，获得 8 个聚类集，对于每个聚类集，去除含有元素个数过少的聚类类别（噪声），本文中，去除元素个数小于总个数 10%的类别。

本文采用 X-means 聚类算法[15]，X-means 是一种 K-means 的改进算法，不需要预先输入类别数，能自动调整至一个合适的类别数，当对目标数据信息的具体分布了解很少时，X-means 是一种非常合适的算法。

$$Q_c = \exp(-\beta \bullet \frac{sd}{c}) \qquad (11)$$

对于每个聚类集的每个聚类类别，利用公式(11)评价聚类质量，sd 为该类别标准差，c 为平均值，β 为预设的常数，一般取 2.5。

本文分析及检测的对象是流序列。每一种僵尸类别都会产生表 1 所示的 8 个聚类集的特征统计信息，如 SDbot、BlackEnergy、Rbot 等不同的僵尸程序会产生不同的统计信息 S_{sd}、S_{be}、S_{rb}。对于一个未知流序列，亦可计算得到对应这 8 个特征的一个向量 T，针对 T 中的每一个特征值 T[i]，分别与 S_{sd}、S_{be}、S_{rb} 等已知的特征模型中对应的特征进行比较，若满足：

$$c-2sd \le T[i] \le c+2sd \qquad (12)$$

则说明该流序列在该聚类类别的接受范围内，则为所属的特征模型加上相应的分数，将 T 的每个特征值和所有特征模型比较之后，选出得分最高的特征模型，如果 T 与该模型的特征匹配次数大于等于 σ_1（σ_1 为预先设定的阈值），且得分大于等于 σ_2（预先设定的得分阈值），则该未知流序列 T 属于该特征模型，则可判定 T 属于该类僵尸。

表 1 SDbot 特征

特征	类别大小	类别均值	类别标准差	Q_c
sinterval(ms)	12	169779.10	7685.52	0.89
	6	136560.08	11038.27	0.81
	10	996.75	6.59	0.98
dinterval(ms)	6	136743.41	11177.47	0.82
	12	170477.03	7147.75	0.90
	10	999.97	7.97	0.98
sduration(ms)	18	0.14	0.27	0.01
	4	132.51	8.42	0.85
	6	249.10	8.65	0.92
dduration(ms)	23	0.29	0.37	0.04
	3	251.09	8.72	0.92
sip_bytes(byte)	19	69.54	7.21	0.77
	4	178.77	6.57	0.91
	3	127.00	0.01	0.99
dip_bytes(byte)	10	40.00	0.01	0.99
	8	77.44	3.92	0.88
	5	103.00	0.01	0.99
	4	126.79	0.37	0.99
freq(Hz)	8	1.99	0.015	0.98
	19	0.014	0.011	0.16
acf_freq(Hz)	8	0.33	0.0026	0.98
	18	0.0053	0.0019	0.40

注：表 1 是 19 个 SDbot 样本产生的 28 个流序列聚类及分析后的特征提取情况，排除了类别个数小于等于 2 的类别，得到的 8 个聚类集，22 个聚类类别。

被公式(12)接受后，加分由以下公式决定：

$$\delta_{score} = Q_c \bullet Q_{Ti} \quad Q_{Ti} = \exp(-\beta \bullet \frac{sd}{c}) \qquad (13)$$

其中 Q_c 为命中类别的聚类质量，Q_{Ti} 是流序列 T 在 i 属性上的质量，sd 为流序列 T 在该属性值上的标准差，c 是在该属性值上的均值，β 仍然取 2.5，即加分取决于聚类质量与流序列 T 本身的质量（即随机程度，T 中的流的特性越随机，质量越低）。值得注意的是，freq 与 acf_freq 是无法通过计算标准差的方法描述质量的，在本文中，freq 的 Q_{Tfreq} 取值是 $Q_{Tinterval}$（综合 $Q_{Tsinterval}$ 与 $Q_{Tdinterval}$ 得出的流时间间隔综合质量），Q_{Tacf_freq} 的取值是公式(9)计算出的 ω。

如一个未知流序列 T 的属性值向量是(170000,

140000, 250, 250, 180,103, 1.99, 0.33)，对于源流间隔时间 170000，假设质量为 0.88，分别比较不同的特征模型 S_{sd}、S_{be}、S_{rb} 中的 sinterval 特征，在 S_{sd} 中，170000 被第一个聚类类别 169779.10±7685.52*2 接受，则为 score_S_{sd} 加上 0.98*0.88 并为 $match\_count_{sd}$ 加 1；对于向量中每一个属性值都做这样的比较，选出最高得分 score_S_{max}，假设对应特征模型是 S_{sd}，若 score_S_{sd} 大于等于阈值 σ_2 且 $match\_count_{sd}$ 大于等于阈值 σ_1，则判定 T 是 SDbot 流序列。

4 实验分析

4.1 系统实现

系统结构如图 2 所示，运行于 linux 操作系统下，高性能组流模块的输入是离线 pcap 文件，或者在线的实时报文数据。利用 libpcap 抓包后，进行多线程组流，根据传输层协议的不同，为 TCP、UDP、OTHER 创建不同的线程同时进行，可极大的提升组流效率。

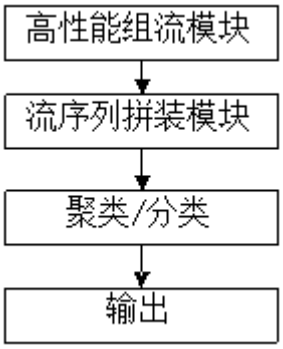

图 2 系统结构图

报文分发时为三种传输层协议创建循环队列，由于读写各只有一个线程，循环队列可以无锁。组流模块输出流记录文件（即记录流五元组、开始时间、结束时间、报文数、字节数、结束标志等流的统计信息）。

流序列拼装模块输入是流记录文件，将流及其逆向副本按照（sIP、dIP、dPort）为标识，组装成流序列，并利用快速傅里叶变换、循环自相关等方法计算出该流序列所有的 8 项属性值。如果输入的是已知僵尸程序的流记录，则进行 3.3 节所述聚类分析，并输出特征文件；如果输入是未知流记录，则进行 3.3 节所述分类分析，并将检测出的僵尸流记录输出。

4.2 已知僵尸程序特征分析

本文从已有蜜罐系统及各类僵尸程序源码中获取僵尸程序样本。一共分析了 4 种僵尸程序累计数十个僵尸样本，SDbot（IRC）、Shadowbot（IRC）、BlackEnergy（HTTP）、Zbot（HTTP）。其中 SDbot 产生于 2002 年，是最经典的 IRC 僵尸程序之一；Shadowbot 类似于 SDbot，是一个简单的 IRC 僵尸程序；BlackEnergy 是一种经典的 HTTP 僵尸程序，主要用于 DDOS 攻击；Zbot 是著名的恶意僵尸程序工具包 ZEUS 产生的僵尸程序，基于 HTTP 协议。

为获得纯净的僵尸通信流量，本文通过 Anubis 分析各个 bot 样本启动后在操作系统中的各项行为，为防止僵尸样本影响正常网络，本文在虚拟机环境下单独运行，并构建相应 IRC 服务器与 web 服务器，利用 wireshark 获取各个样本产生的网络通信流量。

表 2 本文使用的僵尸程序

僵尸程序类型	僵尸样本数	产生的流序列数
SDbot	19	56
Shadowbot	10	40
BlackEnergy	10	38
Zbot	23	46

注：表 2 所示是训练所用数据的统计情况，对于每一种僵尸程序类型，都会产生如表 1 所示的特征表，用于对新流序列的检测。

4.3 交叉验证

本文将表 2 所示流序列的一半用于特征训练，另一半用于验证。用于验证的流序列将和大约 80G 的正常流量混合，正常流量来自于实验室一台用作报文转发的服务器自 2013 年 3 月 15 日至 2013 年 3 月 25 日的数据，这台服务器转发的流量一般用作日常科研、网页浏览及部分的 P2P 下载，基本可以保证这 80G 流量不包含僵尸流量。同时为避免噪声流量的干扰，本文只关注流个数在 16（10-20 均可）的流序列。

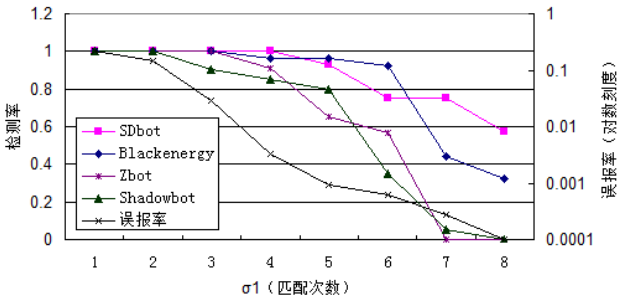

图 2 检测效果与 σ_1 的关系

需要确定的阈值有两个，一个是匹配次数最小值阈值 σ_1，另一个是匹配得分阈值 σ_2。图 2 描述的是各类 bot 的检测率和整体误报率随 σ_1 变化的情况，确定 σ_1 值的时候 σ_2 暂时设定为 0。根据图 2 的实验结果，本文取 σ_1 为 5，即当 8 个特征中匹配至少 5 个时，才有可能被判定为僵尸流量。当 σ_1 为 5 时，正常流量中大约只有 0.1%的 σ_1 能达到 5 及以上，而僵尸样本流量在 σ_1 为 5 时平均仍保持有 75%以上的检测率。再结合 σ_1 的判定，可使检测率保持在一定水平的前提下，获得很低的误报率。

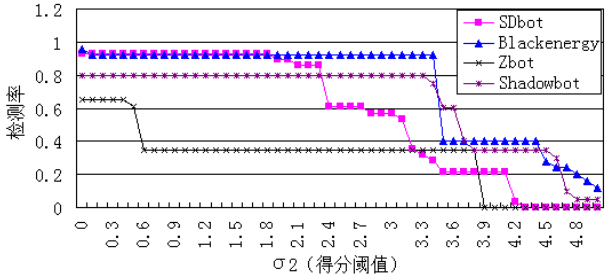

图 3 四类 bot 检测效果比较（σ_1=5）

图 3 是四类 bot 分开检测时获得的检测效果随着 σ_2 变化情况的对比，可以看出，SDbot、Blackenergy 以及 Shadowbot 在 σ_1=5，σ_2<2.0 时，都保持有很高的检测率，而 Zbot 的检测率从 σ_2>0.5 开始，检测率就下降到 50%以下，据对 Zbot 僵尸流量的人工分析，得知 Zbot 流序列中流的分布并不均匀，而是部分短流穿插于长流之中，导致流序列 Q_T 在流时间间隔、流持续时间等属性上的质量很低，即使命中，获得的 δ_{score} 很低，致使总得分 σ_2 较低。对于 σ_2 的取值主要依据图 4 的实验结果，有 96 条僵尸流序列混入大约 80G 的正常流量中，在 σ_1=5 的前提下，有 11 条流序列没有被检测出，大多来自于 Zbot 流序列，有 3 条流序列被误报为其他僵尸序列，分别是 SDbot 被检测为 Zbot，Shadowbot 被检测为 Blackenergy，Shadowbot 被检测为 SDbot。

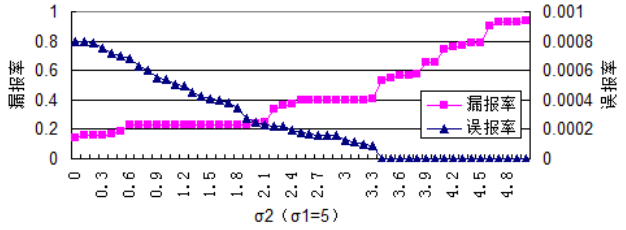

图 4 系统整体检测效果

根据图 4 的结果，本文取 σ_2 为 2.0，可使误报率在 0.02%左右的情况下获得 75%-80%的检测率（20%-25%的漏报率）。

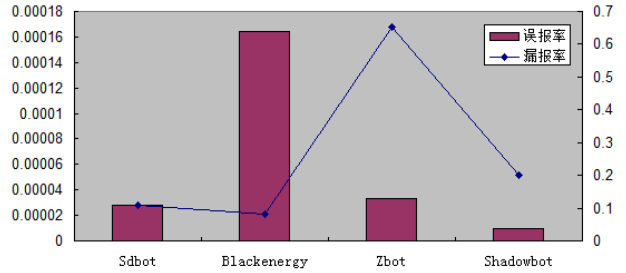

图 5 各类 bot 误报率/漏报率对比

图 5 是各类僵尸误报率和漏报率的对比，误报率 Blackenergy 最高，Shadowbot 最低，主要是因为 Blackenergy 作为 HTTP 僵尸程序，流序列特征主要是规律性的 HTTP 的请求，在大量的正常流量情况下，也有一定的概率会出现这种特征。而同为 HTTP 僵尸的 Zbot，由于其 HTTP 请求流序列的特殊性（短流分布于长流），误报率则较低。漏报率 Zbot 最高，并且明显高于其他三种，原因如上文所提，即 Zbot 流序列中长短流交替所导致。

5 总结

本文通过研究 SDbot（IRC）、Shadowbot（IRC）、BlackEnergy（HTTP）、Zbot（HTTP）四种僵尸客户端与 C&C 服务器交互时产生的流量特性，以流序列为研究对象，针对性的提取流间隔时间、流持续时间、流字节数、交互频率等特征，采用一维聚类及相应的评分方式，建立僵尸模型并与未知流量进行匹配，从而达到检测僵尸网络的目的，取得了良好的效果；并在此基础上，实现了一个小型的僵尸网络检测系统。

本文实现的系统仍有不少需要改进的地方，首

先，对于 Zbot 这样的长短流交替的僵尸流序列，本系统的检测率较低；其次，本文选取的僵尸样本有限，很难检测到网络中所有的僵尸流量，如果能找到一个更为通用的建模方法及模型，则有更好的检测效果；最后，本文并没有涉及到 P2P 僵尸网络的检测。

参考文献

[1] Zhaosheng Zhu, Guohan Lu, Yan Chen, et al. Botnet Research Survey[J]. Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International, pp.967-972, July 28 2008-Aug. 1 2008.

[2] Craig A.Schiller, Jim Binkley, David Harley, et al. Botnets: The Killer Web App[M]. 科学出版社, 2009 年 8 月 1 日.

[3] J.Goebel, T.Holz.Rishi. Identify bot contaminated hosts by IRC nickname evaluation[A]. In Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets(HotBots'07)[C], Apr 2007.

[4] 王威, 方滨兴, 崔翔. 基于终端行为特征的 IRC 僵尸网络检测[J]. 计算机学报, 2009 年 10 月, 第 32 卷, 第 10 期.

[5] Konrad Rieck, Guido Schwenk, Tobias Limmer, et al. Botzilla: detecting the "phoning home" of malicious software[J]. In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10).

[6] Jae-Seo Lee, HyunCheol Jeong, Jun-Hyung Park, et al. The Activity Analysis of Malicious HTTP-based Botnets using Degree of Periodic Repeatability[A]. International Conference on Security Technology[C], 2008.

[7] Hyunsang Choi, Heejo Lee, Hyogon Kim. BotGAD: detecting botnets by capturing group activities in network traffic[A]. In Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE (COMSWARE '09)[C], 2009.

[8] Ian Castle, Eimear Buckley. The Automatic Discovery, Identification and Measurement of Botnets[A]. Second International Conference on Emerging Security Information, Systems and Technologies[C], 2008.

[9] Binbin Wang, Zhitang Li, Dong Li,et al. Modeling Connections Behavior for Web-based Bots Detection[A]. 2010 2nd International Conference on e-Business and Information System Security[C], 2010.

[10] Gu G., Zhang J., Lee W.. Botsniffer: Detecting botnet command and control channels in network traffic[A]. Proceedings of the 2008 ISOC Network and Distributed System Security Symposium[C], February 2008.

[11] Guofei Gu, Roberto Perdisci, Junjie Zhang, et al. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection[A], USENIX Security[C], 2008.

[12] Florian Tegeler, Giovanni Vigna, Xiaoming Fu, et al. BotFinder: Finding Bots in Network Traffic Without Deep Packet Inspection[A]. ACM Co-NEXT'12[C], December 10-13, 2012.

[13] G.Gu, P.Porras, V.Yegneswaran, et al. BotHunter: Detecting malware infection through ids-driven dialog correlation[A]. In Proceedings of the 16th USENIX Security Symposium (Security'07)[C], August 2007.

[14] 张琛, 王亮, 熊文柱. P2P 僵尸网络的检测技术[J]. 计算机应用, 第 30 卷增刊 1, 2010 年 6 月.

[15] D. Pelleg, A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters[A]. In Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00)[C], pages 727–734, San Francisco, CA, USA, 2000.