一种抗污染的混合 P2P 僵尸网络

尹 捷[1,2], 崔 翔[1,3], 方滨兴[3,4], 衣龙浩[1,2], 张方娇[1,2]

[1] 中国科学院信息工程研究所, 北京 中国 100093
[2] 中国科学院大学网络空间安全学院, 北京 中国 100049
[3] 广州大学网络空间先进技术研究院, 广州 中国 510006
[4] 电子科技大学广东电子信息工程研究院, 广东东莞 中国 523808

摘要 基于 Peer-list 的混合型 P2P 僵尸网络代表了一类高级僵尸网络形态, 这种僵尸网络的优势是可抵抗传统 P2P 僵尸网络易受的索引污染(Index Poisoning)攻击和女巫(Sybil)攻击, 然而却引入了新的问题——易受 Peer-list 污染攻击。本文提出一种新颖的混合 P2P 僵尸网络设计模型, 在僵尸网络构建和 Peer-list 更新的整个生命周期中引入信誉机制, 使得 Peer-list 污染攻击难以发挥作用。实验证明该模型具备很强的抗污染能力和很高的健壮性, 因此对网络安全防御造成了新的威胁。最后, 我们提出了若干可行的防御方法。本文旨在增加防御者对高级僵尸网络的理解, 以促进更有效的网络防御。

关键词 P2P 僵尸网络; 混合型僵尸网络; 对等列表; 污染攻击
中图法分类号 TP309.5 DOI 号 10.19363/j.cnki.cn10-1380/tn.2018.01.005

A Pollution-resilient Hybrid P2P Botnet

YIN Jie[1,2], CUI Xiang[1,3], FANG Binxing[3,4], YI Longhao[1,2], ZHANG Fangjiao[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
[3] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China
[4] Institute of Electronic and Information Engineering of UESTC in Guangdong, Dongguan Guangdong 523808, China

Abstract Peer-list exchanging based hybrid P2P botnets, which are naturally robust in topology structure and immune to Index Poisoning and Sybil attacks, represent one of the most sophisticated botnets. However, such kinds of botnets are generally vulnerable to Peer-list pollution attack. In this paper, we present a novel hybrid botnet design, which aims to verify the possibility of developing a pollution resilient hybrid P2P botnet. The proposed botnet introduces a reputation-based mechanism into the whole lifecycle of Peer-list constructing and updating, making pollution attack extremely difficult, even using thousands of coordinated polluters simultaneously. We evaluated the proposed botnet under mitigation condition; and the experiments result show that such kind of advanced botnet is feasible, consequently posing a great challenge to security defenders. At last, we suggest some possible countermeasures to defend against such an advanced botnet. The ultimate goal of our work is to increase the understanding of the emerging advanced botnets, which will promote the development of more efficient countermeasures.

Key words P2P botnet; Hybrid botnet; Peer-list; Pollution attack

1 引言

僵尸网络(Botnet)是一种通过入侵网络空间内若干非合作用户终端构建的、可被攻击者远程控制的通用计算平台[1]。其中, 被入侵的“用户终端”称之为僵尸主机(Bot); “攻击者”指掌握僵尸主机资源, 对其具有实际操控权的控制者(Botmaster); “远程控制”指的是攻击者可以通过命令与控制(Command and Control, 简写为 C&C)信道一对多地控制非合作用户终端。僵尸网络作为攻击者手中最有效的攻击平台之一, 被广泛用于恶意软件分发(Malware Distributing)、勒索软件分发(Ransomware Distributing)、垃圾邮件分发(Spam)、分布式拒绝服务攻击(Distributed Denial of Service, DDoS)、钓鱼攻击(Phishing)、用户身份窃取(Massive Online Identity Theft)、点击欺诈(Click Fraud)、电子货币挖掘(Digital

通讯作者: 崔翔, 博士, 研究员, Email: cuixiang@gzhu.edu.cn。
本课题得到国家重点研发计划 No. 2016QY08D1602, 东莞市引进创新科研团队计划(项目编号: 201636000100038)资助。
收稿日期: 2017-09-25; 修改日期: 2017-11-13; 定稿日期: 2017-12-05

Currency Mining)等网络犯罪活动, 成为网络空间安全面临的重大威胁, 引起国际上的广泛关注。

1.1 僵尸网络发展趋势

随着互联网的不断发展, 云计算、移动终端的泛化, 以及创新应用的涌现, 僵尸网络也在不断演变进化。从感染终端来看, 攻击者已经不再局限于传统PC, 而是将目标转向以智能手机、无线路由器、智能家居、IoT 设备为代表的多种终端设备; 从拓扑结构来看, 传统中心结构僵尸网络逐步向去中心化结构发展, 以避免中心结构存在的单点失效(Single Point of Failure)问题; 从 C&C 协议来看, 僵尸网络从早期简单的 IRC 协议, 发展到如今广泛应用且相对隐蔽的 HTTP 协议, 再到更为复杂的 P2P 协议, 其生存能力不断增强; 从 C&C 服务器生存性来看, 攻击者从利用 VPS(虚拟专用服务器)转向利用公共网络服务(如 Twitter、GitHub)甚至通信卫星, 并将 C&C 服务器隐藏在暗网之中, 隐蔽性逐步提高。僵尸网络的发展已然趋向复杂化和多样化, P2P 模式更是因为其良好的可扩展性和健壮性成为僵尸网络的高级形态之一。

1.2 P2P 僵尸网络的脆弱点

针对 P2P 僵尸网络, 根据其通信协议又可分为结构化和非结构化[2]。结构化僵尸网络通常采用基于分布式哈希表(Distributed Hash Table, DHT)的 P2P 协议, 如 Strom 僵尸网络使用基于 Kademlia 算法的 Overnet 协议[3]。这类 P2P 僵尸网络的固有脆弱点是易受索引污染(Index Poisoning)攻击[4]和女巫(Sybil)攻击[5], 易于通过 Crawler 和 Sybil 节点测量其规模[6]。非结构化僵尸网络通常采用随机扫描方式或者节点列表(Peer-list)方式进行通信。基于随机扫描方式的僵尸网络存在流量异常的固有脆弱点; 采用基于 Peer-list 交换方式进行通信, 每个僵尸主机需要维护一份包含节点信息的列表, 即 Peer-list, 并定期随机挑选其中的部分节点进行通信, 获取攻击者指令和更新节点信息, 这类 P2P 僵尸网络具有较好的灵活性、可扩展性、健壮性, 在许多流行的 P2P 僵尸网络中使用, 如 Kelihos[7], Waledac[8], ZeroAccess[9]。然而, 基于 Peer-list 交换的僵尸网络容易受到污染(Peer-list Pollution)攻击。防御者可以部署大量污染节点长期活跃地在僵尸网络中, 频繁地与正常节点交换 Peer-list。因为污染节点的各种行为表现如同正常节点(激进的防御者甚至可允许污染节点参与真正的攻击), 所以难以区分。当污染节点占据了正常节点 Peer-list 的绝大部分位置后(即污染了正常节点的 Peer-list), 全部污染节点在同一时刻停止运行, 就可以达到破坏僵尸网络的目的。知名僵尸网络 Waledac 正是因其协议设计存在漏洞而被微软协同学术机构通过 Peer-list 污染进行关闭[10]。因此, 理想 P2P 僵尸网络设计目标是在保证高度可扩展性和健壮性的同时, 将 P2P 网络易受的各种威胁(包括 Bootstrap 攻击、Index Poisoning 攻击、Sybil 攻击、Peer-list Pollution 攻击)降至最低。

1.3 本文贡献

本文针对基于自定义 Peer-list 交换协议僵尸网络所存在的脆弱点, 提出了一种抗污染的混合型 P2P 僵尸网络模型(Pollution-resilient Hybrid P2P Botnet, 以下简称 PrBot)。PrBot 首次将信誉评估机制引入到僵尸网络构建和 Peer-list 更新的整个生命周期中, 使得传统的针对 P2P 僵尸网络的攻击难以发挥作用; 实验证明 PrBot 具有高健壮性和很好的抗污染能力;最后, 我们从检测、测量、追踪等方面提出了一系列防御措施。类似 PrBot 的新型僵尸网络很可能在未来的网络攻击中出现, 对互联网造成威胁。本文旨在增加防御者对此类僵尸网络的理解, 并探讨有效的防御手段。

2 研究现状

针对新型僵尸网络的分析和预测, 研究人员开展了大量研究工作。Rossow 等人[11]提出了一种形式化攻击图模型, 并使用这个模型评估 11 个知名的 P2P 僵尸网络的特性与防御措施。结果显示部分僵尸网络是容易对抗的, 而另一部分复杂结构的僵尸网络对大多数防御方法都具有高对抗性, 值得安全人员关注。在结构化 P2P 僵尸网络方面, Starnberger 等人[12]提出一种基于 Kademlia 的僵尸网络命令控制信道协议 Overbot, 僵尸主机的请求流量隐藏在合法 P2P 应用中, 难以被追踪和发现, 极大提升了僵尸网络整体健壮性和隐蔽性。Yan 等人[13]提出了一种名为 AntBot 的抗污染结构化 P2P 僵尸网络, Antbot 采用树状结构来传播命令, 即使单个 bot 被捕获也仅暴露有限的信息, 具有随机性和冗余性。Napp 等人[14]利用基于 P2P 的 Skype 协议作为通信载体实现了一种新型僵尸网络, 该方法可穿透防火墙和 NAT 设备, 其通信流量隐藏在合法 Skype 流量之中, 难以被区分和过滤。

在混合型 P2P 僵尸网络方面, Vogt 等人[15]提出了一种混合结构的僵尸网络设计方法, 该设计思想是自动把一个大规模僵尸网络划分成多个僵尸子网, 僵尸子网之间通过 P2P 协议连接起来, 具有彼此独立不易暴露整体规模的优势, 可管理大规模僵尸网

络而不会出现性能瓶颈, 本文采用了不同于该模型的另一种混合型 P2P 结构, 详见第 3 节。Hund 等人[16]提出了一种难以被追踪和关闭的 P2P 僵尸网络 RamBot, 该模型采用基于信誉评分(Credit-point System)的主机验证机制以及工作量证明机制(Proof-of-work)判断通信节点身份的真实性, 能有效对抗 Peer-list 污染以及 Sybil 攻击, 具有较好的主机生存性, 但该工作缺乏对提出模型的健壮性、有效性做出说明和评估。Wang 等人[17]提出了一种混合型 P2P 僵尸网络, 控制者通过特定的 Senor 主机来收集网络中节点信息, 并随机选择可用节点组成新的 Peer-list, 发送给每个请求 Peer-list 更新的主机。该方法可用以实现负载均衡, 同时平衡了网络中节点的连接性, 具有较好的健壮性, 但是并没有解决节点间的信任关系问题, 仍然易受到 Peer-list 污染攻击。不同于上述已有工作, 本文受到文献[17]的启发, 提出了一种新型 Peer-list 更新方法, 并引入信誉评估机制, 兼顾了僵尸网络的健壮性和抗污染能力。

3 PrBot 模型设计

3.1 两类 Bot 程序

文中 PrBot 即可代表整个僵尸网络, 也可代表僵尸程序, 其含义取决于上下文。PrBot 分为两类: Servant Bot(以下简称 Servant)和 Client Bot(以下简称 Client)[17]。

定义 1 **Servant**: 指具有静态公网 IP 地址, 且可接受远程客户端主动发起的连接并与之通信的公网可达主机。

定义 2 **Client**: 指具有内网 IP、动态 IP 或者虽然具有静态公网 IP, 但却因防火墙拦截而无法接受由外至内的网络访问的非公网可达主机。Client 不会出现在 Peer-list 中, 但每个 Client 与 Servant 一样, 都必须包含 Peer-list。

PrBot 感染新主机后, 首先需要判定自身类型是 Servant 还是 Client。判定方法是开放一个服务并请求 Peer-list 中其他 Servant 尝试访问该服务, 如果可以成功访问, 说明自身可作为 Servant, 否则标记自身为 Client, 然后将身份标签写入本地可持续存储。判定过程如下:

1) 新感染主机获取自己的 IP 地址列表, 考虑到主机可能同时有多个网络连接(如以太网连接、Wi-Fi 连接、4G 网连接、ADSL 连接), 因此可能同时具有多个内外网 IP。将获取的 IP 地址列表记为<IP_1, IP_2, …, IP_n>;

2) 在 IP 地址列表中删除内网 IP 和动态 IP (可利用 Whois 查询工具)。注意, 这里可能会删除潜在的 Servant, 因为有的内网 IP 经过外部路由器映射后也可以充当服务器使用。尽管如此, 为了简单起见, 依然删除此类 IP 地址;

3) 选择一个本系统尚未监听的且对防火墙穿透性强的 TCP 端口(如 80、443、21), 并在所有公网 IP 地址上监听, 监听的 TCP 端口记为 P;

4) 从 Peer-list 中随机选择若干当前在线的 Servant, 将 IP 地址列表及监听的端口信息发送给每个 Servant, 请求 Servant 执行连接;

5) 如果检测到 Servant 可以连接到自身某个 IP, 则认为该 IP 是本机的公网可达 IP, 记为 IP_i, 并标记自身为 Servant。如果不存在这样的 IP, 则标记自身为 Client。如果有多个 IP 地址同时满足条件, 则随机选择一个。

3.2 命令与控制信道

PrBot 的整体 C&C 体系设计如图 1 所示, 是由基于混合 P2P 结构的主信道(Hybrid P2P-based C&C, 简称 HPCC)与基于 URL Flux 协议的失效备份信道(URL Flux-based C&C, 简称 UFCC)共同构成。

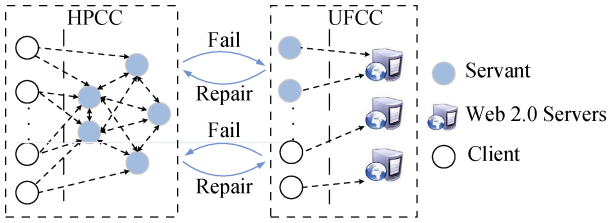


图 1 PrBot C&C 体系

Figure 1 PrBot C&C Architecture

HPCC 的上层由 Servant 构成整个 P2P 僵尸网络骨干网, 如果将骨干网看作是一个服务器群, 下层 Client 则是大量的客户端。如图 2 所示, Client D 的 Peer-list 包含 Servant A 和 B, Client C 的 Peer-list 包含 Servant A 和 E, Servant A 的 Peer-list 包含 Servant B、E、F。就命令发布而言, PrBot 拓扑是无向图, 并提供两种基本的信息传递模式: 推模式(Push)和拉模式(Pull)。推模式指控制者主动将命令推送给僵尸主机, 当僵尸主机接收到之前从未见过的命令时, 则把该命令转发到它能联系到的其他主机; 拉模式是僵尸主机主动向 Peer-list 中的 Servant 发起请求, 询问命令。获得命令后, 僵尸主机将根据命令执行指定活动。在命令传输过程中, PrBot 采用对称秘钥和非对称密钥进行信息加密和身份认证, 防止被防御人员监控和劫持。密码学在僵尸网络中的应用已经非常成熟, 本文不再赘述。

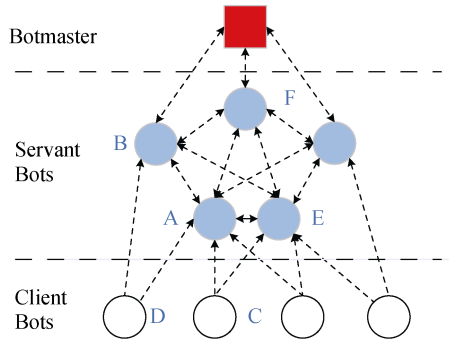


图 2　混合 P2P C&C 结构
Figure 2　Hybrid P2P-based C&C

HPCC 虽然具有良好的健壮性, 但其初始化方法依然依赖于硬编码的 Peer-list, 存在 Bootstrap 失效问题。为避免该问题, PrBot 采用了基于 URL Flux 协议[18]的备份信道。一旦 HPCC 失效, UFCC 将被激活, 僵尸主机通过该信道可以直接获取由控制者发布的可信 Peer-list, 用以重构 HPCC, 当 HPCC 恢复后, UFCC 则再次转入休眠状态。

UFCC 利用互联网上免费的 Web 2.0 服务(例如, Twitter, GitHub, Pastebin, TinyURL)作为 C&C 信道实现资源动态定位。其本质是控制者和僵尸程序共享同一套 URL 生成算法(URL Generation Algorithm, UGA), 控制者将签名后的有效 Peer 信息发布到互联网上, 并生成一个可被僵尸程序预测的 URL 地址, 僵尸程序按照预定的算法生成 URL 地址池逐一进行访问, 从而获取新 Peer 以构建新的 Peer-list。如果 UFCC 失效, 控制者可以通过 HPCC 更新 UGA 算法进行恢复。该备份信道可以灵活的利用众多免费的 Web 2.0 服务[19], 图 3 利用了文本存储网站(例如, Pastebin)和缩址服务(例如, TinyURL)来实现资源动态定位, 更多其他的方法可参考文献[20, 21]。

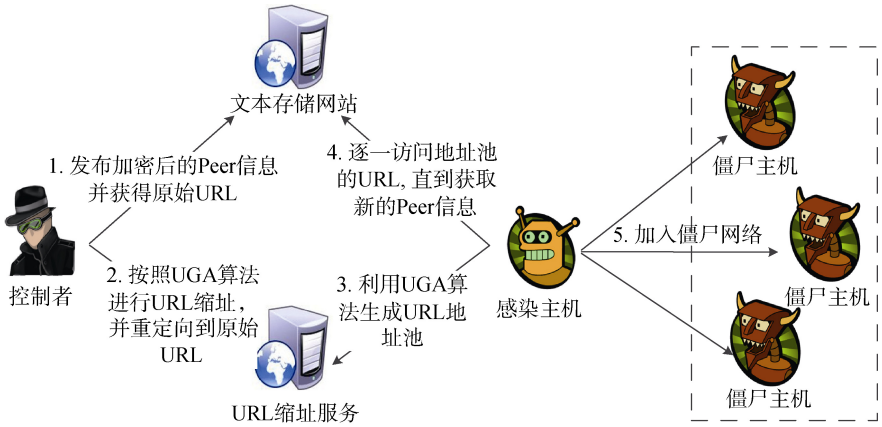


图 3　基于 URL Flux 协议的 C&C
Figure 3　URL Flux-based C&C

UFCC 这种失效恢复的思路看起来是存在矛盾的, 如果确信 URL Flux 是有效的获取控制信息的方法, 那为什么还需要设计混合 P2P 结构? 原因有两方面: (1)安全风险因素: UFCC 从原理上来讲依然存在单点失效问题, 一旦 URL 失效就无法获取信息, 在扩展性、控制信息发布简单性方面不如混合型 P2P 网络。(2)对抗成本因素: HPCC 与 UFCC 共同作用一定程度上增加了防御者的对抗成本。

3.3　Peer-list 格式

定义 3 Peer-list: 由若干条 peer 记录组成, 每条记录描述一个 Servant 信息, 记为<ServantIP, ServantPort, BotID, FailCount>。Peer-list 是 Peer 记录的集合, 假设 B_i 表示僵尸主机, M 为 Peer-list 大小, 则 B_i 的 Peer-list 可表示为:

$$\left\{\begin{array}{l}\langle IP_1, P_1, BotID_1, FC_1\rangle \\ \langle IP_2, P_2, BotID_2, FC_2\rangle \\ \cdots \\ \langle IP_M, P_M, BotID_M, FC_M\rangle\end{array}\right\} \tag{1}$$

其中, ServantIP(IP)表示该 Servant 的 IP 地址; ServantPort(P)表示该 Servant 开放的 TCP 端口; BotID 用来在整个僵尸网络内唯一地标识自身。由于互联网上广泛部署的 DHCP 和 NAT 设备使基于 IP 的身份识别并不准确, 因此控制者可以使用 BotID 代替 IP 地址标识僵尸主机, 从而提高对僵尸网络控制和测量的准确性; FailCount(FC)表示僵尸主机 B_i 与该 Servant 交换控制信息的累计失败次数。

FC 的作用是评估节点(即僵尸主机)的信誉值。FC 值越高, 其可信度越低, 在 Peer-list 更新过程中,

具有高 FC 值的节点将优先被替换掉。引起该值增加的原因包括但不限于网络连接失败, 网络连接成功但获取信息失败。当 FC 达到预设的最大值后, 该节点将会直接从 Peer-list 中删除。出现以下两种情况时, 节点的 FC 将直接到达最大值, 触发删除操作:

1) **提供虚假信息。**PrBot 采用公/私钥对来进行身份认证, 控制者用私钥将命令进行签名, 僵尸程序用公钥验证, 一旦发现命令不是来自控制者, 则 FC 达到最大值, 将节点从 Peer-list 中删除。

2) **未通过蜜罐检测验证。**控制者需要定期探测所感染的终端是否为防御者部署的控制环境, 例如通过设计一个 *Detect* 命令, 收到 *Detect* 命令的节点需要收集自己 Peer-list 中 Servant 节点的相关信息, 一旦不符合标准, 则 FC 达到最大值, 并从 Peer-list 中删除该节点。由于篇幅关系, 本文列出以下 3 条检测标准, 但不限于此:

• 终端用户的行为分析。请求节点汇报本机行为, 比如是否经常使用某些软件, 是否长时间在线等。

• 感染终端软硬件环境。请求节点汇报本机软硬件环境和资源, 比如操作系统版本、计算能力、带宽等。

• 攻击能力证明。出于法律和道德约束的考虑, 防御人员部署的蜜罐节点不会产生攻击流量危害真实互联网, 不少蜜罐会对向外发出的恶意流量进行屏蔽或速率限制, 控制者可以通过僵尸主机对自主建立的感知节点进行攻击测试, 并收集攻击流量, 从而评估节点的真实性[23]。

4 对抗条件下的 PrBot 构建

4.1 P2P 僵尸网络对抗模型

针对 P2P 僵尸网络, 常用的对抗手段包括 Index Poisoning 攻击、Sybil 攻击、Peer-list Pollution 攻击。基于 DHT 思想的结构化 P2P 僵尸网络易受到 Index Poisoning 的威胁, 通过掌握僵尸主机命令搜索算法, 防御者可以提前向索引中注入伪造信息来干扰和阻断命令传递。Sybil 攻击方法指防御者将单个节点伪装成多重身份加入到僵尸网络中, 由于缺少中心认证机构, 一旦网络中正常的僵尸主机访问到 Sybil 节点, 防御人员可以利用这些 Sybil 节点对 P2P 僵尸网络实施规模测量、活动监视、请求中断和欺骗。Davis 等人[6]利用 Sybil 攻击对 Storm 实施了拒绝服务攻击。基于 Peer-list 的非结构化 P2P 僵尸网络往往容易遭受 Peer-list 污染, 该种对抗方法与 Sybil 攻击十分相近, 其基本思想是在网络中部署 Polluter 节点并向僵尸主机传递大量其他恶意节点或者虚假的节点地址, 利用 Peer-list 更新机制占据列表空间, 使僵尸主机无法访问到合法节点, 导致僵尸网络被分割或脱离控制。

混合 P2P 结构的僵尸网络, 能免疫 Index Poisoning 攻击、Sybil 攻击。图 4 描绘了混合 P2P 僵尸网络对抗模型, 假设 A 和 A1 是僵尸主机, B 和 C 是易感染主机, P 是 Polluter 节点, H 是蜜罐节点。针对 Peer-list Pollution 攻击, 通常使用两种主要策略。

1) **部署 Polluter 节点**。在逆向分析僵尸网络通信协议之后, 防御者可以定制僵尸程序, Polluter 节点运行防御者定制的僵尸程序, 其各种行为与正常节点相同, 但不参与真正的攻击。Polluter 节点需要长期活跃在僵尸网络中, 通过频繁地与正常节点交换 Peer-list, 使自己进入正常节点的 Peer-list 中。当 Polluter 节点占据了正常节点 Peer-list 中绝大部分位置后(即污染了正常节点的 Peer-list 表), 全部 Polluter 节点在约定的时刻骤然停止运行, 可以破坏 P2P 僵尸网络。

2) **部署蜜罐节点**。防御者可以通过部署蜜罐节点来捕获僵尸程序, 从而进一步分析僵尸网络的通信行为, 了解攻击方法, 辅助部署 Polluter 节点。此外, 当僵尸程序感染了蜜罐节点后, 防御者可以伪造通信消息, 使蜜罐节点活跃在僵尸网络中, 并占据在正常节点的 Peer-list 中, 起到与 Polluter 节点相同的作用, 进而联合发起污染攻击。

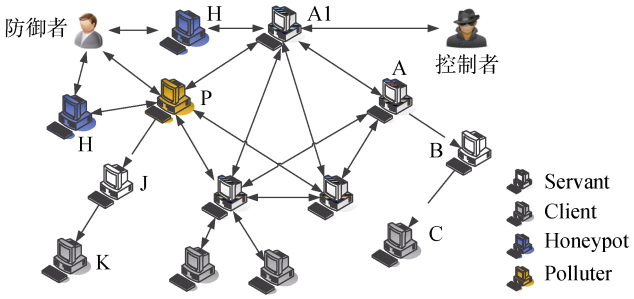


图 4 混合 P2P 僵尸网络对抗模型

Figure 4 Hybrid P2P botnet model under mitigation condition

4.2 PrBot 构建

4.2.1 感染

僵尸程序可以利用多种传播方法, 比如钓鱼邮件, 文件资源共享, 漏洞利用, 蠕虫等[24]。为了描述简洁, 本文假设 PrBot 是通过漏洞利用进行传播。被感染的僵尸主机首先会通过对内、外网主机扫描, 定位脆弱主机资源, 然后利用漏洞投递载荷并运行, 从而实现感染。

定义 4 感染源: 若存在僵尸主机 A1 感染了脆弱主机 A, 则 A1 是 A 的感染源。

定义 5 InfectNum: 记录本机感染脆弱主机的数量。InfectNum 值越低说明该节点感染的主机数越少, 对僵尸网络所做的贡献就越少。

新感染主机上的僵尸程序一定已经携带(硬编码)了一个 Peer-list, 该 Peer-list 或者来源于控制者的初始化, 或者来自于感染源经过若干次更新与交换后生成的最新 Peer-list。通过感染传递 Peer-list 的过程如图 5 所示, 其中红色记录代表感染源。此时, 如果新感染僵尸主机判定自身为 Client, 则将其感染源的信息添加进自身的 Peer-list 中, 并替换 FailCount 值最高的一条记录, 如果存在多条记录 FailCount 相等且最大, 则随机选择一个进行替换, 并进行感染源标记。同时将自己的信息, 如 BotID, IP 地址, 主机状态发送给感染源, 并周期性地访问 Peer-list 的节点以获取命令。如果僵尸程序判定自身为 Servant, 除了将感染源加入 Peer-list 中, 还需要通知其他节点执行 Peer-list 更新过程(详见 4.2.3)。需要注意的是, PrBot 在感染过程中遵守**感染源可信定理**。

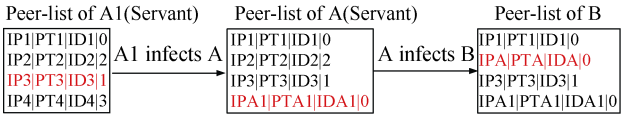


图 5　Peer-list 传递过程

Figure 5　Peer-list delivery via infection

定理 1 感染源可信定理: 感染源是可信的。

证明: 反证法。假设 A1 是不可信的, 即 A1 是防御者部署的蜜罐或者 Polluter 节点。当 A1 感染 A 时, A 将得到由 A1 所提供的僵尸程序以及 Peer-list (可能含有更多 Polluter)。当 A 继续感染主机 B 时, B 也会信任主机 A 以及由 A 提供的 Peer-list(包含 A1)。至此, 防御者利用 A1 已经成功污染了僵尸主机 A 和 B 上的 Peer-list, 因为 A 和 B 的 Peer-list 来源于 A1 且包含 A1。当 A1 联合其他 Polluter 节点突然关闭时, A 和 B 将会受到影响, 甚至无法继续寻址。然而, Polluter 节点能够污染的主机, 都是因该 Polluter 节点造成的, 如图 4 中的 P->J->K 的感染链所示。所以, 防御者所做工作不仅没有对真正僵尸主机形成污染, 反而产生了额外的危害, 更不用说法律和道德问题。因此, 可以完全信任感染源以及由感染源所提供的信息, 这不会对僵尸网络本身有负面影响。

4.2.2　重复感染

当僵尸主机再次被感染时, 称为重复感染(Re-infection)。感染源可能是曾经的感染源, 也可能是其他新的主机。一般来说, 重复感染可以有效地将不同的感染路径互连在一起, 使得僵尸网络更加健壮[15], 但同时, 这也带来了一定的风险, 因为防御者可以部署 Polluter 节点感染那些已经被感染过的节点, 从而监控和污染整个僵尸网络。因此, 重复感染源并不都是可信的。为了对抗防御者的污染策略, PrBot 不考虑重复感染情况, 当僵尸主机再次被感染时, 并不会复制该重复感染源的 Peer-list。

4.2.3　Peer-list 更新机制

对于防御者来说, 由于感染与重复感染的限定, 污染 PrBot 的唯一机会就是利用 Peer-list 的更新过程。因此, 确保 PrBot 抗污染能力是 Peer-list 更新机制的设计核心。从健壮性来看, 随着僵尸网络逐渐扩张, 大量新的僵尸主机将会加入, 为了避免初始硬编码的 Peer-list 被大量新加入的节点复制, 导致单点失效问题, 网络中的节点需要对自己的 Peer-list 进行不断更新, 以平衡节点的度, 保持网络拓扑的稳定。然而, 如果所有节点都将新加入的节点更新到 Peer-list 中, 则该新节点就会被大量连接, 成为超级节点, 这不满足混合 P2P 结构的要求。因此, 为了避免超级节点的产生, 需要控制执行更新操作的节点数量, 本文提出了一种自动广播更新机制, 通过设定两个广播跳数(*hop_x*, *hop_y*)来限定需要进行更新操作的节点数量, 以确保网络的健壮性。同时, 将 FC 和 InfectNum 引入更新过程, 以提高抗污染的能力。

算法 1. Peer-list 更新算法

算法的输入: {*update, <info>, hop_x(y)*}

update: 表示 Peer-list 更新命令, 任何收到 *update* 命令的节点都要做出相应的更新操作。

<info>: 表示新感染的节点信息, 即<ServantIP, ServantPort, BotID, FailCount>

hop_x: 广播跳数, 用于向 Peer-list 中感染源广播时使用。

hop_y: 广播跳数, 用于向 Peer-list 中除感染源以外的节点广播时使用。

hop_x 和 *hop_y* 是两个整型数值, 初始化值被硬编码在程序内。在节点广播时携带, 每经过一次广播, 其值减 1, 当为 0 时, 广播停止。假设 A1 是僵尸主机 A 的感染源, A[2…M]是 A 的 Peer-list 中除感染源以外的其他节点的集合, B 是易感染主机。Peer-list 更新过程如图 6 所示, 描述如下:

1) 当僵尸主机 A 感染主机 B 时, 如果 B 是 Servant, 则 A 开始启动更新过程。首先, A 把更新指令广播给 Peer-list 中的所有节点, 记为 A{*update,*

<*info*>, *hop_x*}->A1, A{*update*, <*info*>, *hop_y*}->A[2…M]。然后，将 B 的信息(即<*info*>)添加进 Peer-list 中。添加时需要判断 Peer-list 的情况：若 A 的 Peer-list 未满，则将<*info*>加入到空记录里；若 Peer-list 已满，则删除当前具有最高 FC 的一条记录(多条相同最高 FC 记录，则随机选择一条)替换为<*info*>；若该节点已经存在(根据 IP 地址判断)，则不再添加。

2) 当 A1 收到{*update*, <*info*>, *hop_x*}指令后，首先将 *hop_x* 值减 1，并将指令广播到自己的感染源(假设为 A11)，记为 A1{*update*, <*info*>, *hop_x-1*}->A11。然后生成新的初始 *hop_y* 值，并广播给除 A11 之外的其他节点(假设为 A1[2…M])，记为 A1{*update*, <*info*>, *hop_y*}->A1[2…M]。最后，根据 Peer-list 中的情况添加<*info*>。

3) 当 A[2…M]中各节点收到{*update*, <*info*>, *hop_y*}指令后，首先将 *hop_y* 值减 1，并将指令广播给自己 Peer-list 中所有节点(假设为 A[2…M][1…M])，记为 A{*update*, <*info*>, *hop_y-1*}->A[2…M][1…M]。然后，判断自己的 InfectNum 是否为 0，如果为 0，则根据 Peer-list 中的情况添加<*info*>，否则不做更新。

4) 所有被广播的节点按照接受到的指令做出相应的更新操作(参照 A1 和 A[2…M])。当 *hop_x* 和 *hop_y* 值为 0 时，则停止广播。

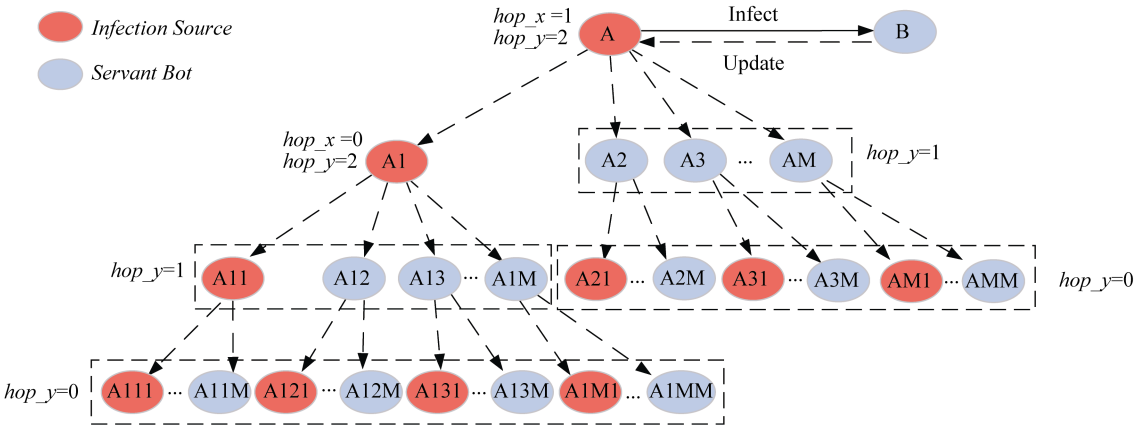


图 6 Peer-list 更新过程

Figure 6 Peer-list updating procedure

4.3 安全分析

4.3.1 健壮性分析

Peer-list 更新机制中，*hop_x* 和 *hop_y* 是用来控制需要进行更新的节点数量，使网络结构保持稳定状态。一个新节点加入，如果进行全网广播，则该节点将可能被大量其他节点更新到 Peer-list 中，而变成超级节点，降低了网络的健壮性。一个新节点的加入，理论上接收到更新指令的节点数量可表示为

$$N_{receive} \approx (hop\_x+1)M^{hop\_y} \quad (2)$$

但考虑到 InfectNum 的限制，以及存在重复转发、节点不在线的可能，可用ξ(0<ξ<1)表示受限节点比率，则实际接收到更新指令的节点数量可表示为

$$N'_{receive} \approx \xi(hop\_x+1)M^{hop\_y} \quad (3)$$

此时，$N'_{receive} < N_{receive}$。在我们的建模实验中，当 M=20，Servant 节点为 5000 时，选取 *hop_x*=1，*hop_x*=2，最多有 800 个节点将收到有关新加入节点的更新指令，即 $N_{receive} \approx 800$，大约有 300 个节点会将该新节点加入到 Peer-list 中(不考虑节点不在线的情况)，即 $N'_{receive} \approx 300$。假设该新节点是防御者部署的蜜罐节点，则意味着约有 300 个节点将添加该节点到 Peer-list。即便如此，PrBot 的 Peer-list 更新机制可以使网络不依赖某个或某些节点，当蜜罐节点停止转发命令，网络仍然具有较强的连通性，并不会受到影响。我们将在第 5 节中对 PrBot 的健壮性进行详细的评估。

采用 P2P 结构的僵尸网络具有很好的健壮性，但由于僵尸主机之间需要频繁通信并进行 Peer-list 更新，因此会引起流量异常。然而，中心化结构的僵尸网络同样存在流量异常。基于 IRC 协议的僵尸网络可以通过监测传输层流量来检测；基于 HTTP 协议的僵尸网络虽然可以混合在正常流量之中，但健壮性不好；基于 Domain-Flux、Fast-Flux 协议的僵尸网络具有较好的健壮性，但其动态寻址的特点会产生明显的流量异常，已有较多研究方法来检测此类僵尸网络[39, 40]。因此，保证僵尸网络的健壮性的同时，不可避免地会引起流量异常，影响其有效性，在防御机制成熟的网络中可能会被发现，进而被清理。

4.3.2 抗污染性分析

在对抗条件下, PrBot 具有很好的抗污染能力。Peer-list 更新过程中 FC 和 InfectNum 的判定分别降低了 Polluter 和蜜罐节点在僵尸网络中的生存性和有效性, 使污染 Peer-list 难以实现。即使部分节点被污染, 由于网络具有很强的健壮性, 仍然可以有效抵抗污染攻击。

1)对于 Polluter 节点来说, Peer-list 更新过程是自发性的, 这就使得 Polluter 节点无法主动推送自己, 只有当节点感染了新的主机才会进行更新。但考虑到多个 Polluter 节点可以联合进行相互广播的情况, PrBot 定义了 InfectNum 来限制受污染节点的数量, 保护高权重的节点。网络中节点的权重 Q 可表示为

$$Q \propto \frac{K}{N}(InfectNum+1) \tag{4}$$

其中, K 为节点的度, N 为网络中节点总数。节点的权重正比于节点的度 K 和感染数量 InfectNum, 感染数量和节点的度越高, 权重则越高, 对网络的作用及影响就越大。假设新加入的节点是 Polluter 或者蜜罐, 根据 Peer-list 更新机制, 只有 InfectNum 为 0 的节点才会添加它们, 而这些节点在僵尸网络中的权重较小, 即使被污染, 对整个网络的影响并不大。

2) 对于蜜罐节点来说, 除了 InfectNum 的限制, FC 的设置使得蜜罐节点必须保持高的可信度和活跃度, 并且还需通过蜜罐检测验证, 否则 FC 值将会增加甚至达到最大, 面临被替换和删除的危险。信誉机制被应用于 Peer-list 构建与自动更新的整个生命周期中, 提高了僵尸网络的抗污染能力。

5 实验评估

5.1 相关参数与评估指标

考虑到真实环境下的实验受限, 本文采用基于 Python 的网络建模, 模拟僵尸网络的构建, 包括节点加入、删除、更新等操作, 并对 PrBot 的健壮性和抗污染能力进行了评估。下面给出五个评估指标:

1) 度分布[25]: 一个节点的度通常定义为该节点连接的所有边的总和, 网络的度分布 $P(K)$即为网络中节点的度的频率分布, 其中 K 表示节点的度。

2) 连通率(*Connected Ratio*)[17]: 表示随机删除 p 个 Servant 节点后, 剩下的节点能够相互连通的能力。假定僵尸网络中节点数为 N, 删除 p 个节点后, 最大联通子图中的节点个数为 N_d, 则连通率可表示为

$$C(p)=\frac{N_d}{N-p} \tag{5}$$

$C(p)$值越高, 说明僵尸网络可连通节点数越多。

3) 度率(*Degree Ratio*)[17]: 表示随机删除 p 个 Servant 节点后, 剩下的节点聚集在一起的密集程度。度率可表示为

$$D(p)=\frac{\text{去点后最大连通子图中节点的平均度}}{\text{去点前原始网络中节点的平均度}} \tag{6}$$

$D(p)$值越高, 说明节点之间连接得越紧密。

4) 可达率(*Reached Ratio*): 表示随机删除 p 个 Servant 节点后, 控制者发布的命令成功到达节点的比率。可达率体现了 Botmaster 对僵尸网络控能力的强弱。假定僵尸网络中节点数为 N, 删除 p 个 Servant 节点后, 控制者随机选择 20 个节点发出指令, 接收到指令的节点个数为 N_r, 则可达率可表示为

$$R(p)=\frac{N_r}{N-p} \tag{7}$$

$R(p)$值越大说明可达率越好。

5) 时效性(*Command Efficiency*): 表示随机删除 p 个 Servant 节点后, 控制者发布的命令到达各节点的平均延迟。时效性体现了僵尸网络对命令控制的反应速度, 但命令到达的延迟与僵尸主机的性能、在线情况、网络带宽及拥堵情况等诸多因素有关, 具有较强的偶然性。在理想的条件下, 本文使用平均跳数来衡量命令时效, 并设命令每经过一个节点的检索时间 T=60s, 假定僵尸网络中规模为 N, 删除 p 个 Servant 节点后, 控制者随机选择 20 个节点发出指令, 通过 i 跳接收到指令的节点数量为 W_i, 所需的最大跳数为 h, 则时效性可表示为

$$T(p)=\frac{\sum_{i=1}^{h} W_i \cdot i}{N-p}T \tag{8}$$

本文对 PrBot 进行建模所用参数可表示为 φ(N, M, f), 其中, N 指僵尸网络的规模; M 指 Peer-list 的大小; f (0 <f <1)表示 Client 和 Servant 的百分比, 因此 Servant 的数量是 N*f。为了便于后续与文献[17]提出的模型进行比较, 本文假设在模拟环境下有 500,000 台易受感染的脆弱主机, 当僵尸网络的规模达到 20,000 时停止增长, 即 N= 20,000, M=20。

f 值的选取需要考虑多种因素, 因为许多 Client 是位于防火墙或 NAT 设备内, 还有一些采用动态地址分配。正如文献[11]指出的, 一般来说, 僵尸网络中大约有 60%~87%的设备不能接收连接请求。不少研究人员也对此做了测量和评估, Kang 等人[26]提出了一个 Passive P2P Monitor (PPM)模型, 即使主机在防火墙或者 NAT 设备内, 也能够将其枚举出来。他们通过该模型测量了 Storm 僵尸网络, 其结果显示大

约有 40%的僵尸主机是在防火墙或者 NAT 设备内。为了更好地理解 Servant 节点对整个僵尸网络影响，我们取 f = 0.25, 0.5, 0.75 来对 PrBot 的 $C(p)$和 $D(p)$值做评估。

假设 $\varphi(20{,}000, 20, f)$, $hop\_x = 1$, $hop\_y = 2$, 评估结果如图 7(a)和(b)所示。不难看出，随着 Servant 的比例增加, $C(p)$和 $D(p)$值也会增长，这是因为 Servant 节点形成了整个僵尸网络的骨干网，相互连接，不断更新。Servant 的数量越多，则 PrBot 的健壮性越好。当 Servant 节点去掉了 80%，在 f = 0.75 的情况下，其 $C(p)$仍然稳定在 1，而在 f = 0.25 的情况下, $C(p)$与 $D(p)$都已经降低到 0.4 以下。考虑到现实世界中存在着大量的 Client，在本文的后续评估实验中，选取 f = 0.25 作为评估参数。

表 1　$\omega(hop_x, hop_y)$参数设置

Table 1　$\omega(hop\_x, hop\_y)$ settings

ω	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
hop_x	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
hop_y	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3

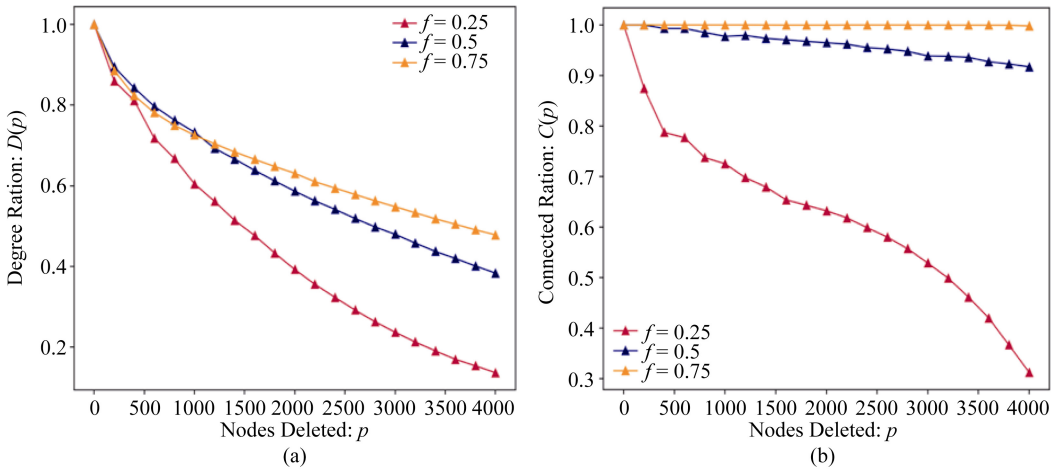


图 7　在 f = 0.25,0.5,0.75 的条件下的 $D(p)$、$C(p)$比较

Figure 7　The $D(p)$ and $C(p)$ comparison under f = 0.25,0.5,0.75

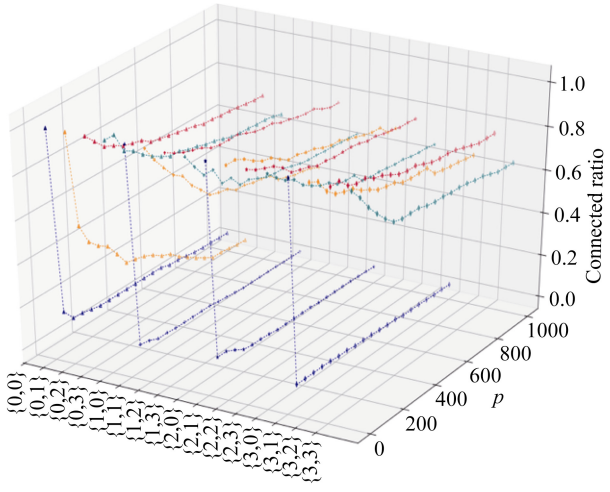


图 8　$\omega(hop_x, hop_y)$不同取值下的 $C(p)$比较

Figure 8　The $C(p)$ comparison under different ω (hop_x, hop_y) settings

第 4 节中提到, hop_x 和 hop_y 对于 PrBot 的健壮性和抗毁性很重要，它们决定了当感染一个新主机时，会有多少个已存在的节点更新自己的 Peer-list。为了更好地理解广播机制对僵尸网络的影响，并且找到最优 hop_x 和 hop_y 值，我们选取了不同的参数值 ω(hop_x, hop_y)(见表 1)进行模拟评估。

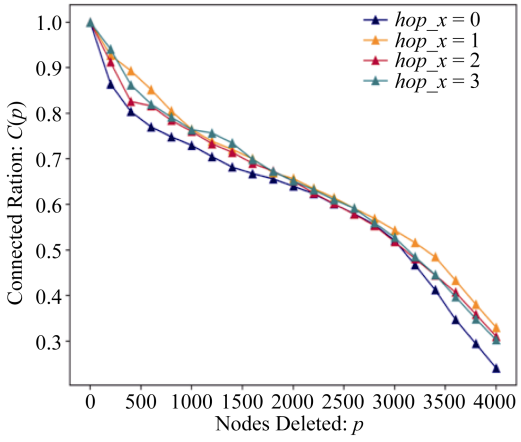


图 9　当 hop_y = 2 时, hop_x 不同取值下的 $C(p)$

Figure 9　The $C(p)$ under different hop_x settings when hop_y=2

***hop_y* 最优值选择**: 如图 8 所示, 四条蓝线代表 *hop_y*=0 时的情况, 其 *C*(*p*)值接近最底部 0 处, 这意味着网络中几乎没有执行 Peer-list 更新过程。初始 Peer-list 中的节点充当了中心服务器, 被大量新节点连接, 如果被移除, 网络将受到严重打击。四条红线代表了 *hop_y*=2 时的情况, 此时, *C*(*p*)稳定在 0.8 左右, 是几种取值下表现最好的情况, 因此可以得出 *hop_y*=2 为最优取值。

***hop_x* 最优值选择:** 图 8 不能直接观测不同 *hop_x* 值所带来的变化, 因此, 我们在 *hop_y*=2 条件下重新评估 *hop_x* 对网络连通率 *C*(*p*)的影响, 结果如图 9 所示, 当 *hop_x*=1 时, 僵尸网络的 *C*(*p*)值最高, 健壮性更好, 因此 *hop_x*=1 为最优值。

5.2　健壮性评估

为了评估 Peer-list 更新机制对僵尸网络健壮性的影响, 我们比较了其他两种 Peer-list 更新方式。

Case1: 该 Peer-list 更新方式是由 Wang 等人[17]在所提出的一种高级混合僵尸网络中所采用的, 当僵尸网络中 Servant 的数量达到 1000 时, 控制者通过 Sensor 节点来收集信息, 并执行一次 Peer-list 更新过程。控制者随机从这 1000 个 Servant 中选取节点组成一个新的 Peer-list, 然后发送给每个向它请求 Peer-list 更新的僵尸主机。

Case2: 僵尸网络中不执行 Peer-list 更新, 当新感染节点加入时, 只有感染源将其添加到自己的 Peer-list 中。

实验共模拟包括 PrBot 在内的三种 Peer-list 更新机制的僵尸网络构建过程, 三种方法对应的实验场景采用相同参数 φ (20,000,20,0.25), 对于 PrBot, 额外使用 $hop\_x = 1$ 和 $hop\_y = 2$。

三种僵尸网络的度分布如图 10 所示(为方便作图, 将节点的度取对数)。黄色线条代表 Case1, 可以看出, 用作更新的 1000 个 Servant 节点的度从 300 到 500 不等, 而在更新之后所加入的新节点的度大约在 20 到 30 之间, 这 1000 个 Servant 节点形成了该僵尸网络的骨干网, 被大部分其他节点连接。蓝色线条代表 Case2, 初始 Peer-list 中节点的度大约在 14,000 到 17,000 之间, 而其他节点的度非常小, 可以看出整个网络处在不平衡链接状态。这些初始节点在传播过程中, 不断被复制, 形成了超级节点, 完全充当了中心服务器的角色。红色线条代表 PrBot, 从线条的分布来看, 网络中节点的度分布均衡, 所有节点的度大约从 50 到 400 不等, 网络中不存在超级节点, 任意移除部分节点, 对整个网络的影响不会致命。另外注意, 这里的节点指的是 Servant 节点, Client 节点的度总是为 M, 因为它自己 Peer-list 中的 Servant 节点通信。与 Case1 和 Case2 比较, PrBot 具有更好的健壮性。

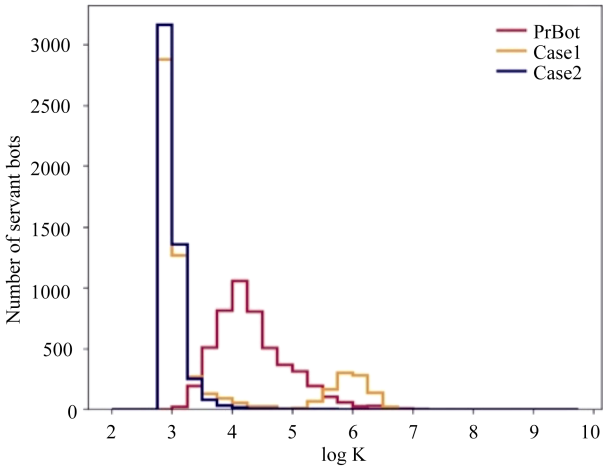


图 10　度分布比较图

Figure 10　The degree distribution comparison

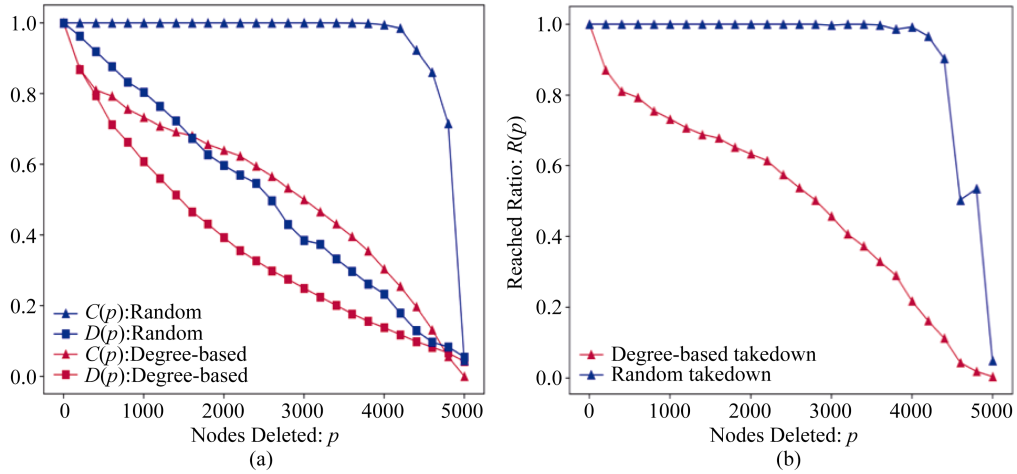


图 11　两种去点策略的 *C*(*p*)、*D*(*p*)、*R*(*p*)比较

Figure 11　The *C*(*p*), *D*(*p*) and *R*(*p*) comparison under two takedown strategies

5.3 抗污染评估

假设防御者在僵尸网络中部署了大量的污染节点(只作为 Servant), 当污染节点占据了正常节点 Peer-list 中绝大部分位置后, 如果全部污染节点在特定的时刻停止运行, 便可以降低 PrBot 的可用性, 甚至摧毁整个僵尸网络。我们通过移除 Servant 节点, 来模拟该攻击。本文采用两种去点策略[27]:

随机去点(Random Takedown): 在网络中随机选择 Servant 节点并移除。

按度去点(Degree-based Takedown): 按节点的度由高到低进行去点。

评估结果如图 11(a)、11(b)、图 12 所示。从图 11(a)中我们可以看出, 在随机去点的情况下, 如果移除的 Servant 节点不超过 80%, 则网络仍然处于高连接状态($C(p) > 0.98$)。在按度去点的策略下, 如果超过一半的 Servant 节点被移除, 则整个僵尸网络将会受到较为严重的打击, $C(p) < 0.5$、$D(p) < 0.3$。当所有 Servant 节点都被移除时, 整个网络将会崩溃, $C(p) = 0$、$D(p) = 0$。

图 11(b)和图12 分别显示在两种去点策略下的可达率和时效性。在随机去点策略下, 只要低于 80%的 Servant 被移除, 则每一个节点都能收到命令($R(p) = 1$), 而转发跳数大约在 2.5 左右, 命令到达各节点的时效大约为 150s。红色曲线代表了按度去点策略下的情况, 显然地, PrBot 在随机去点的策略下具有更好的连通性能。事实上, 防御者是很难控制污染节点的度, 因为 Peer-list 的更新过程是完全自动的, 因此在随机去点策略下的性能更具有实际参考意义。实验结果说明即使在防御者部署了大量污染节点的情况下(不超过 80%), PrBot 仍可以正常运作, 具有较强的抗污染能力。

为了进一步说明, 同其他僵尸网络模型相比, PrBot 具有明显的抗污染能力的优势, 我们选取了 Case1 作为比较对象。评估结果如图 13(a)和图 13(b)所示。在 Case1 中, 当 Servant 节点数量达到 1000 时, 进行一次更新操作, 采用按度去点策略。图 13(a)显示了两种模型下的 $C(p)$和 $D(p)$比较情况。显然地, 对于 Case1 来说, 度最高的节点是这 1000 个 Servant 节点, 并且它们将组成新的 Peer-list, 如果去除这些节点, 意味着新的 Peer-list 将无效, 整个僵尸网络将被瓦解, $C(p)$和 $D(p)$趋近于 0。对于 PrBot 来说, 其 Peer-list 更新机制是自发且有限的, 即使度最高的 1000 个节点是污染节点, 也不会产生较大的影响, 因为随着僵尸网络的不断增长, 节点的 Peer-list 会不断的进行自我更新, 即使是去掉它们, 仍然具有较好的连通率和度率, $C(p)$值约为 0.76, $D(p)$值约为 0.62。图 13(b)是两种模型下 $T(p)$的比较, PrBot 的连通跳数稳定在 2 左右, 命令到达各节点的时效约为 130s, 而 Case1 起伏较大, 体现了被污染后网络结构处在不稳定状态。值得说明的是, 对于中心结构僵尸网络, 僵尸主机通常不经过其他节点(或者经过少量中

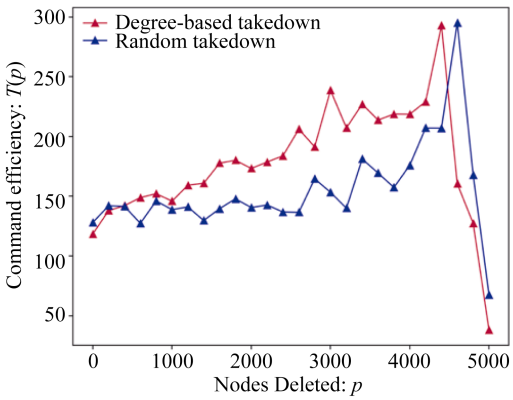


图 12 两种去点策略的 $T(p)$比较

Figure 12 The $T(p)$ comparison under two takedown strategies

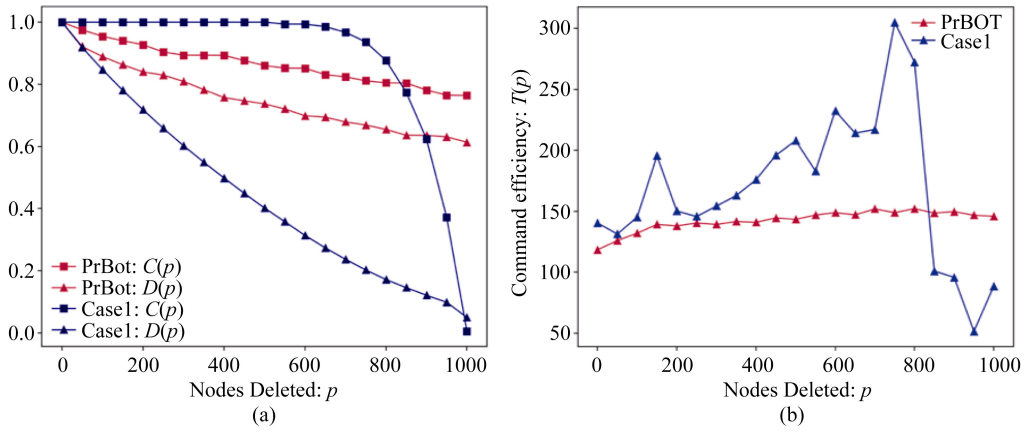


图 13 两种模型下 $D(p)$、$C(p)$、$T(p)$比较

Figure 13 The $D(p)$, $C(p)$ and $T(p)$ comparison under two cases

转节点)访问命令控制服务器, 路径数通常较短。因此, 中心结构的效率一般要比 P2P 结构的高, 本文不再进行比较。

6 防御

针对僵尸网络防御的研究, 主要从对抗、检测、测量、追踪这四个方面开展。其中, 对抗的目的是接管僵尸网络控制权或降低其可用性; 检测的目的是发现新的僵尸网络; 测量的目的是刻画僵尸网络拓扑结构、活跃规模、完全规模等可度量属性及其动态变化轨迹, 展现出僵尸网络的轮廓和特征; 追踪的目的是获知僵尸网络的内部活动。针对 PrBot 这类高抗性的僵尸网络, 其防御策略可从检测、测量、追踪三方面入手。

6.1 检测技术

僵尸网络的检测方法通常可分为基于蜜罐分析、基于通信内容、基于异常、基于数据日志、基于挖掘分析五类。

基于蜜罐的分析方法首先通过蜜罐诱捕的办法获取大量恶意代码样本, 在可控环境中进行监控分析, 发现僵尸程序及其恶意行为。虽然 PrBot 能够有效抵抗污染攻击, 但是防御者仍然可以通过部署蜜罐节点来捕获僵尸程序, 通过分析其行为特征, 有助于进一步了解僵尸网络通信机理及潜在脆弱点, 制定下一步关闭和打击僵尸网络的策略。

基于通信内容的检测是普遍采用的防御方法, 通过事先配置特征匹配规则, 诸如 Snort 等传统入侵检测系统可快速、准确发现僵尸网络。该方法适用于具有明确特征的僵尸网络, 缺点是无法检测未知的僵尸网络, 需要持续维护和更新特征码知识库。因此, 如果防御者获知了 PrBot 通信中备份 C&C 中的 URL 生成算法, 则可通过配置 URL 规则来检测可疑的通信内容。

基于异常检测方法主要围绕僵尸网络引发的异常进行检测, 具体又可分为主机层异常和网络层异常。主机层异常检测通过监控终端行为发现异常, 监控内容包括进程、文件、注册表、网络连接等行为[28]。网络层异常检测则假定僵尸程序与控制服务器之间的通信模式较正常用户通信具有较大差异性, 从而可通过流量分析来发现僵尸网络的踪迹。网络流量分析系统可部署在网络边界上实时收集网络数据流(类似 NetFlow 格式), 也可汇总不同网络的多个路由器发送来的 NetFlow 流, 常见的异常特征包括高网络延迟、高流量、非常规端口流量等[29]。PrBot 由于其 Peer-list 广播更新机制, 会存在高流量等异常的行为, 基于网络层异常的检测方法可以适用。同时, PrBot 具备僵尸网络的基本属性, 即处在同一僵尸网络中的主机具有时空相似性, 并执行类似的恶意活动, 防御者可以部署 EDR(end-point detection and response)、NDR(network detection and response)、或者基于该类属性的检测系统[30, 31]来进行全面地检测和响应。

基于数据日志的分析检测方法与基于异常检测的方法比较相似, 但数据源大多来自 DNS、邮件等日志记录。研究人员通过分析日志中的异常来识别可能的僵尸网络行为[32, 33]。例如, 若控制者利用 PrBot 发送垃圾邮件, 则可通过分析邮件服务器中账号激活、登录日志、邮件内容相似性等来发现僵尸网络。

基于挖掘分析的检测方法一般借助机器学习算法来进一步识别 C&C 流量。通过对流量聚合分析、多重日志文件关联等来进行识别和检测[34]。Gu 等人[35]认为有相同恶意行为并具有相似通信模式的僵尸主机都可认为是僵尸主机, 并基于该假设提出了一种基于异常检测系统 BotMiner, 该系统首先对数据流按目的地址和端口进行划分, 经过流量属性聚类和主机异常行为关联分析后检测出可疑的僵尸网络通信。该方法具有协议无关性, 可检测加密的僵尸网络, 同样适用于 PrBot。

6.2 测量技术

测量 P2P 僵尸网络时可能会受到合法 P2P 节点、DHCP、NAT、防火墙、开关机等因素的影响。实践证明, 针对同一僵尸网络采用不同的测量方法, 产生的结果可能相差一个数量级[36]。所以, 测量结果必须辅之以特定环境、特定策略、特定方法等的上下文说明才有参考意义。针对 PrBot, 根据 Peer-list 的交换机制可以借助基于 Crawler 思想来测量。防御者通过编写 Crawler 渗透进入僵尸网络, 并模仿真实僵尸程序的 C&C 协议主动联系 Peer, 记录该 Peer 返回的其他 Peer 地址后, 重复执行上述操作, 从而可遍历整个 P2P 网络。在 PrBot 中, 只有 Client 可以主动地向其感染源请求 Peer-list, 如果防御者部署 Crawler 作为 Client 节点, 则可以周期性地从其感染源处请求 Peer-list, 获取部分节点信息。如果作为 Servant 节点, 则可以被动的接收其他节点广播的新节点。当僵尸主机 Peer-list 中无可用节点, 则会启动备份 C&C 信道, 一旦防御者成功的逆向 URL 的生成算法, 那么可以编写 Crawler 执行与正常节点相同的请求, 来获取新的可信节点。虽然以上方法很难精确地评估整个僵尸网络的规模, 但是可以尽可能多的

估算节点数量。

6.3 追踪技术

追踪的前提是掌握 C&C 协议, 基于所掌握的 C&C 协议, 任何僵尸网络都无法避免被追踪。一般所采用的追踪方法可归纳为两类: (1)以渗透的方式加入到僵尸网络中以求掌握僵尸网络内部活动情况, 这种渗透的行为主体称之为僵尸网络渗透者(Infiltrator); (2)在可控环境中运行 Sandbot。基于 Sandbot 思想的追踪是在可控环境中运行 Sandbot, 并对其通信内容进行审计, 从而可获知僵尸网络的活动[37]。但这种方法存在一些弊端, 一方面可控环境可能放行了恶意攻击流量、封锁了必要的 C&C 流量, 也可能错误地篡改了关键内容, 导致被控制者察觉。该方法难以处理加密协议, 对于严格检查运行环境的僵尸程序以及严格检查僵尸程序来源的控制服务器, Sandbot 方法可能会失效。由于 PrBot 设计了检测运行环境的模块, 因此不适合使用 Sandbot 思想的追踪。

PrBot 可以采用基于 Infiltrator 思想的追踪, 该方法是通过模仿真正僵尸程序的 C&C 协议来渗入僵尸网络, 从而可以收到控制者和僵尸程序发来的控制信息, 并获取信息源的地址。Infiltrator 是一种确定的、可控的追踪方法, 只要 Infiltrator 正确模拟了真实僵尸程序的 C&C 协议, 就可以与其他僵尸程序获得相同的来自控制者的控制信息[38]。与真实僵尸程序不同的是: Infiltrator 收到控制命令后, 不会产生相应的攻击动作, 仅在必要时反馈伪造的执行结果。与 Polluter 不同的是, Infiltrator 需严格遵守原有 C&C 协议并进行自我保护, 如果 Infiltrator 的行为偏离了僵尸程序应有的轨道, 则会引起控制者察觉进而招致攻击。而 Polluter 的目的是占据正常节点 Peer-list 中的位置。

以上三种防御技术相辅相成、相互补充, 帮助完善针对 PrBot 的防御体系。除此之外, 防御者还可以采取协同 CERT、ISP 来加强公共服务安全以及封锁端口的防御手段。这主要是因为 PrBot 的备份 C&C 协议依赖于 Twitter, GitHub, 网盘等 Web 服务。Web 服务提供商可以使用用户验证或 CAPTCHA, 甚至使用限速来防止批量注册。这些方法一定程度上可以防止公共服务被滥用。由于逃避检测的技术存在(例如逃避 CAPTCHAs), 因此可考虑多种限制因素, 包括通过访问请求的速率、模式和关联以及用户信誉来检测具有某些组行为的僵尸账号等[15]。

7 结论

本文介绍了一种混合型 P2P 僵尸网络——PrBot。PrBot 采用基于信誉的 Peer-list 更新机制, 可自动化地平衡网络连接, 在具有高健壮性的同时, 还拥有抗污染能力。实验结果也证明 PrBot 能有效对抗防御者实施的大规模协同防御策略。面对新的挑战, 本文从检测、测量、追踪三个方面提出了针对 PrBot 的防御策略。我们认为, 从攻击者角度研究如何构建高对抗性的僵尸网络, 并先于攻击者提出有效的防御策略, 具有重要的现实意义。下一步工作中, 我们将对该类型的僵尸网络进行深入研究, 寻找并设计出快速有效的检测系统。

参考文献

[1] 方滨兴, 崔翔, 王威. 僵尸网络综述[J]. 计算机研究与发展, 2011, 48(8): 1315-1331.

[2] Wang P, Wu L, Aslam B, et al. A systematic study on peer-to-peer botnets[C]//Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on. IEEE, 2009: 1-8.

[3] Holz T, Steiner M, Dahl F, et al. Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm[J]. LEET, 2008, 8(1): 1-9.

[4] Liang J, Naoumov N, Ross K W. The Index Poisoning Attack in P2P File Sharing Systems [A]. // Proceedings of the 25th International Conference on Computer Communications [C], Piscatawary, NJ: IEEE, 2006: 1-12.

[5] Douceur J R. The sybil attack[C]//International Workshop on Peer-to-Peer Systems. Springer, Berlin, Heidelberg, 2002: 251-260.

[6] Davis C R, Fernandez J M, Neville S, et al. Sybil attacks as a mitigation strategy against the storm botnet[C]//Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on. IEEE, 2008: 32-40.

[7] P.-M. Bureau. Same Botnet, Same Guys, New Code: Win32/ Kelihos. In VirusBulletin.

[8] B. Stock, M. Engelberth, F. C. Freiling, and T. Holz. Walowdac – Analysis of a Peer-to-Peer Botnet. In Proceedings of the European Conference on Computer Network Defense.

[9] J. Wyke. ZeroAccess, 2012. Technical Report by SophosLabs.

[10] Dittrich, D.: So You Want to Take Over a Botnet. In: Proceedings of the 5th USENIX Conference on Large-Scale Exploits and Emergent Threats, 2012.

[11] Rossow C, Andriesse D, Werner T, et al. Sok: P2pwned-modeling and evaluating the resilience of peer-to-peer botnets[C]//Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013: 97-111.

[12] Starnberger G, Kruegel C, Kirda E. Overbot: a botnet protocol based on Kademlia[C]//Proceedings of the 4th international conference on Security and privacy in communication netowrks. ACM,

2008: 13.

[13] Yan G, Ha D T, Eidenbenz S. AntBot: Anti-pollution peer-to-peer botnets[J]. Computer networks, 2011, 55(8): 1941-1956.

[14] Nappa A, Fattori A, Balduzzi M, et al. Take a Deep Breath: A Stealthy, Resilient and Cost-Effective Botnet Using Skype[C]// DIMVA. 2010: 81-100.

[15] Vogt R, Aycock J, Jacobson Jr M J. Army of Botnets[C]//NDSS. 2007.

[16] Hund R, Hamann M, Holz T. Towards next-generation botnets[C]//Computer Network Defense, 2008. EC2ND 2008. European Conference on. IEEE, 2008: 33-40.

[17] Wang, P., Sparks, S., Zou, C.C.: An advanced hybrid peer to peer botnet. In: Proceedings of the First Workshop on Hot Topics in Understanding Botnets, HotBots 2007 (2007).

[18] Xiang C, Binxing F, Lihua Y, et al. Andbot: towards advanced mobile botnets[C]//Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats. USENIX Association, 2011: 11-11.

[19] Artturi Lehtiö, F-Secure, Finland. C&C-as-a-service: abusing third-party web services as C&C channels, 2015.

[20] Lee S, Kim J. Fluxing botnet command and control channels with URL shortening services[J]. Computer Communications, 2013, 36(3): 320-332.

[21] Xiang C, Binxing F, Jinqiao S, et al. Botnet triple-channel model: Towards resilient and efficient bidirectional communication botnets [M]. Berlin: Springer, 2013: 53-68.

[22] Wang P, Wu L, Cunningham R, et al. Honeypot detection in advanced botnet attacks [J]. International Journal of Information and Computer Security, 2010, 4(1): 30-51.

[23] Wang P, Wu L, Cunningham R, et al. Honeypot detection in advanced botnet attacks [J]. International Journal of Information and Computer Security, 2010, 4(1): 30-51.

[24] 李可, 方滨兴, 崔翔, 等. 僵尸网络发展研究[J]. 计算机研究与发展, 2016, 53(10): 2189-2206.

[25] West D B. Introduction to graph theory[M]. Upper Saddle River: Prentice hall, 2001.

[26] Kang B B H, Chan-Tin E, Lee C P, et al. Towards complete node enumeration in a peer-to-peer botnet[C]//Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. ACM, 2009, 23-34.

[27] Yen T F, Reiter M K. Revisiting botnet models and their implications for takedown strategies [A]. // Proceedings of the First International Conference on Principles of Security and Trust [C], Berlin: Springer, 2012: 249-268.

[28] Stinson E, Mitchell J C. Characterizing bots' remote control behavior [M]. Berlin: Springer, 2007: 89-108.

[29] Silva S S C, Silva R M P, Pinto R C G, et al. Botnets: A survey [J]. Computer Networks, 2013, 57(2): 378-403.

[30] Zhang J, Saha S, Gu G, et al. Systematic mining of associated server herds for malware campaign discovery[C]//Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on. IEEE, 2015, 630-641.

[31] Gu G, Zhang J, Lee W. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic[C]//NDSS. 2008, 8: 1-18.

[32] Zhuang L, Dunagan J, Simon D R, et al. Characterizing Botnets from Email Spam Records [J]. LEET, 2008, 8(1): 1-9.

[33] Zhao Y, Xie Y, Yu F, et al. BotGraph: Large Scale Spamming Botnet Detection [A]. // Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation [C], Berkeley, CA: USENIX Association, 2009: 321-334.

[34] Seungwon Shin, Zhaoyan Xu, Guofei Gu. "EFFORT: Efficient and Effective Bot Malware Detection." In Proc. of the 31th Annual IEEE Conference on Computer Communications (INFOCOM'12) Mini-Conference, Orlando, Florida, March 2012.

[35] Gu G, Perdisci R, Zhang J, et al. BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection[C]//USENIX Security Symposium. 2008, 5(2): 139-154.

[36] Rajab M, Zarfoss J, Monrose F, et al. My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging [C] //Proc of the first conference on First Workshop on Hot Topics in Understanding Botnets. Berkeley, CA: USENIX Association, 2007:5.

[37] Kreibich, C. and Weaver, N. and Kanich, C. and Cui, W. and Paxson, V. GQ: Practical Containment for Measuring Modern Malware Systems. In Proc. of the 2011 ACM SIGCOMM conference on Internet Measurement Conference. 2011.

[38] Cho C. Y, Caballero J, Grier C, Paxson V and Song D. Insights from the Inside: A View of Botnet Management from Infiltration [C] //Proc of the 3th USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms and more. Berkeley, CA: USENIX Association, 2010: 2.

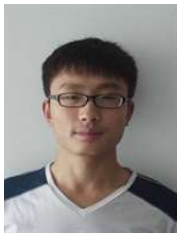
[39] Holz T, Gorecki C, Rieck C, and Freiling F. C. Detection and mitigation of fast-flux service networks [C] //Proc of the 15th Annual Network and Distributed System Security Symposium. Berkeley, CA: USENIX Association, 2008.

[40] Antonakakis M, Perdisci R, Nadji Y, et al. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware[C]// USENIX Security Symposium. 2012: 491-506.

尹捷 于 2014 年在北京联合大学自动化专业获得学士学位。现在中国科学院大学网络空间安全专业攻读硕士学位。研究领域为网络安全、网络攻防技术。研究兴趣包括: 僵尸网络、Web 安全。
Email: yinjie@iie.ac.cn

崔翔 于 2012 年在中国科学院计算技术研究所信息安全专业获得博士学位。现任广州大学网络空间先进技术研究院研究员。研究领域为网络攻防技术。 Email: cuixiang@gzhu.edu.cn

方滨兴 于 1989 年在哈尔滨工业大学计算机系获得博士学位。现任中国电子信息产业集团首席科学家, 中国工程院院士。研究领域为大数据、计算机网络与信息安全。Email: fangbx@cae.cn



衣龙浩 于 2016 年在大连理工大学软件工程专业获得学士学位。现在中国科学院大学网络空间安全专业攻读硕士学位。研究领域为系统安全。研究兴趣包括: 网络安全。
Email: yilonghao@iie.ac.cn

张方娇 于 2014 年在北京邮电大学计算机与科学技术专业获得硕士学位。现任中国科学院信息工程研究所助理研究员。研究领域为网络攻防技术。研究兴趣包括: 恶意代码分析。Email: zhangfangjiao@iie.ac.cn