DOI:10.11918/201904033

# D-BitBot:比特币网络双向通信的P2P僵尸网络模型

周安民[1],钟 毅[1],左 政[2],张 磊[1]

(1.四川大学 网络空间安全学院,成都 610065;2.四川大学 电子信息学院,成都 610065)

**摘 要:** 公有区块链网络(如比特币、以太坊等)具有匿名、难以被关闭的特点,被用于僵尸网络的通信模型研究中,但现有研究中的方法存在网络扩展代价高和回传通道易被溯源的问题.针对上述问题,本文提出D-BitBot,一种基于比特币网络双向通信的点对点(P2P)僵尸网络模型构建方法.该方法使用比特币测试网络作为回传信道,可有效降低数据回收的成本和网络扩展的代价,且能提高回传信道抗溯源的能力;为解决传统僵尸网络上线方式的单点故障缺陷,本文提出一种基于比特币区块链的节点上线机制;另外,为抵御路由表节点注入攻击和僵尸网络节点爬取,本文提出一种基于IP地址加盐哈希排序的节点列表交换算法.实验结果表明,在仿真环境中的D-BitBot上线率达到100%,且具有良好的鲁棒性;在节点请求和节点爬取测试中,本文所提出的算法能有效抵御路由表节点注入攻击和降低现有爬取算法的节点发现率.最后,本文基于3个不同的层面提出可能的抵御方式,并针对本文采用信道的鲁棒性对进行相应的分析和论述.

**关键词:** 僵尸网络;比特币;区块链;P2P;网络仿真

**中图分类号:** TP311.13 **文献标志码:** A **文章编号:** 0367-6234(2020)05-0066-09

# D-BitBot: a P2P duplex botnet model in Bitcoin network

ZHOU Anmin[1], ZHONG Yi[1], ZUO Zheng[2], ZHANG Lei[1]

(1. College of Cybersecurity, Sichuan University, Chengdu 610065, China;
2. College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China)

**Abstract:** Botnets choose blockchain networks (such as Bitcoin and Ethereum) as the communication channel for their command-and-control (C&C) mechanism because blockchain networks are anonymous and hard to shut down. Recent research focuses on this mechanism, but the research methods have such defects that the scalability is restricted, and the upstream channel is vulnerable to existing tracing techniques. To solve these problems, D-BitBot, a peer-to-peer (P2P)-based duplex botnet model which utilizes the Bitcoin testnet as the upstream channel is proposed in this paper. The C&C channel used in this model is hard to trace and reduces the cost of data recovery and network scalability. To avoid single point of failure in traditional botnet bootstrap procedure, a Bitcoin blockchain based bootstrap mechanism is presented. Further, to defend against direct routing table poisoning and P2P botnet crawling, a novel peer list exchange algorithm based on the sorted hash values of IP addresses and random salt values is proposed. According to the result of P2P simulation, D-BitBot provided robust network connectivity with an online rate of 100%. In the node request and node crawling algorithm, the proposed algorithm was effective against direct routing table poisoning and reduced the node detection rate of the current crawling algorithm. Lastly, possible countermeasures and the robustness of the proposed C&C channel were discussed at the end of this paper.

**Keywords:** botnet; Bitcoin; blockchain; P2P; network simulation

僵尸网络是被远程攻击者(Botmaster)所操控的,由一定数量的被攻陷主机(如个人电脑、移动电话或智能设备等)组成的网络,常被用于发起各种网络攻击,如信息窃取、发送钓鱼邮件、DDoS攻击、敲诈勒索和数字加密货币挖矿[1]等.相对于其他恶意软件,僵尸程序的特点是具有命令—控制通信机制(Command and Control, C&C).隐秘可靠的C&C通信机制可提高僵尸网络的健壮性.安全研究人员常通过逆向工程和网络流量[2-3]对僵尸网络进行分析,利用蜜罐技术[4]监控僵尸网络,并制定相应的策略对僵尸网络进行打击[5-6].

传统的C&C通信机制如基于IRC协议和基于HTTP服务器的通信均具有单点故障的缺陷.为提高僵尸网络通信的隐蔽性,某些僵尸网络选择社交网络或网络服务作为通信信道,如基于Yahoo邮件服务的Trojan.IcoScript[7]、基于Skype的僵尸网

**收稿日期:** 2019-04-02
**基金项目:** 国家重点研发计划资金(2017YFB0802900)
**作者简介:** 周安民(1963—),男,研究员
**通信作者:** 张 磊,zhanglei2018@scu.edu.cn

络[8]、基于 Twitter 的僵尸网络[9]和基于 Google Docs 的 Backdoor. Markadocs[10]等. 然而，使用这些信道的僵尸网络控制端仍具有单点故障的缺陷.

为避免单点故障的缺陷，P2P 协议被应用于某些僵尸网络. P2P 僵尸网络按是否基于分布式哈希表(Distributed Hash Table，DHT)可分为结构化(如 Storm Worm[11]、Overbot[12]、Ratbot[13]、Hajime[14]等)与非结构化(如 Slapper[15]、Conficker[16]、eBot[17]、参考文献[18]等). 然而，基于 P2P 协议的僵尸网络对路由表投毒和节点爬取的防御比较薄弱. 此外，采用 P2P 结构的僵尸网络，其上线加入网络过程可能具有单点故障的缺陷.

使用区块链网络作为僵尸网络的 C&C 通信信道，可有效避免上述缺陷，因为某些公有区块链网络(如数字加密货币，以比特币[19]、以太坊[20]为代表)具有匿名、难以被关闭的特点. 比特币是一个基于区块链的，全球性的，公开的分布式账本；以太坊是一个开放的区块链平台，允许任何人在平台中建立和使用通过区块链技术运行的去中心化应用. 二者均通过各自的 P2P 网络进行数据同步. Ali 等[21-22]提出使用比特币主网[23]作为僵尸网络的命令分发信道，使用 web 服务器进行数据回收，回传地址通过命令分发信道进行声明. Pirozzi[24]和 Malaika[25]分别提出使用比特币主网和以太坊主网[26]作为恶意程序的通信信道. 然而，受限于比特币和以太坊高昂的价格，Ali 等[21-22]的研究中，其僵尸网络模型没有使用比特币网络进行上行通信，而是采用声明回传地址的方式回收数据，造成回传信道的抗溯源性相对薄弱；而 Pirozzi[24]和 Malaika[25]研究中的通信模型上下行信道均为区块链主网络，使其难以被溯源. 然而，由于通信成本与网络规模成正比，因此其网络扩展性受限.

在 Ali 等[21-22]的研究中，其僵尸网络通信模型的回传信道存在易被溯源的问题，在 Pirozzi[24]和 Malaika[25]的研究中，其通信模型的网络扩展性受限于数字加密货币价格，此外，传统 P2P 僵尸网络存在节点列表被注入和被爬取的风险. 针对上述问题，本文提出 D-BitBot，一种基于比特币网络双向通信的 P2P 僵尸网络模型. 本文具体工作如下：

1)针对基于比特币网络的僵尸网络通信成本过高和网络扩展性受限的问题，提出一种使用比特币测试网络作为回传信道的技术，能极大地降低由于数据回收所产生的经济开销，使网络扩展性不再受限于比特币价格.

2)为解决传统僵尸网络上线方式的单点故障缺陷，提出一种基于比特币区块链的上线机制，上线节点信息被周期性地记录在比特币区块链上. 比特币交易难以被追溯且无法被屏蔽，有效避免僵尸节点因上线节点失效而无法加入网络.

3)针对传统 P2P 僵尸网络中节点列表存在被注入和被爬取的问题，提出一种基于 IP 地址加盐哈希排序的节点列表交换算法. 通过在节点请求算法中加入随机盐值，保证节点被添加至自身节点时的随机性，使攻击者难以有效地对 D-BitBot 进行节点注入；在节点回复算法中，通过随机盐值生成针对特定 IP 地址的特征值，降低基于特征值欺骗的爬取算法效果.

4)在仿真环境中，对本文提出的 P2P 僵尸网络模型进行构建，并通过节点摘除实验对其鲁棒性进行评估. 实验结果表明，D-BitBot 的上线率达到 100%，移除 95% 的通信节点后，僵尸网络的上线率仍达到 90%.

5)通过重复节点请求实验和网络爬取实验评估本文提出的算法对路由表节点注入攻击和节点爬取的抵御效果. 实验结果显示，该算法使所有待添加节点在被添加至节点列表过程中达到随机选取的效果；P2P 网络节点爬取算法 ZeusMilker[27]对本文所构建的仿真网络进行爬取的平均节点发现率不足 60%.

6)最后，本文基于被感染端、比特币网络和网络服务提供商 3 个不同层面提出可能的抵御方式，并对本文采用信道的经济成本和抗溯源性对进行对应的分析和论述.

## 1　D-BitBot 系统架构

为提高网络管理和数据回传的效率，D-BitBot 分为控制端(Botmaster)、普通节点(Normal Bot)和感知节点(Sensor Bot)三部分. Botmaster 为僵尸网络的控制端，用于发布命令和接收回传数据. Normal Bot 和 Sensor Bot 是僵尸网络的被控端.

每个 Sensor Bot 拥有两个被 Botmaster 认可的比特币公私钥对和对应的比特币测试网络地址，用于发送和接收测试网络比特币. Sensor Bot 的比特币测试网络地址上具有未花费的交易(Unspent Transaction Output，UTXO)，因此 Sensor Bot 具有通过测试网络比特币交易将数据回传至 Botmaster 的能力. Normal Bot 不具有被 Botmaster 认可的比特币公私钥对，或者自身拥有的比特币测试网络地址上没有 UTXO，不具有将数据通过比特币测试网络回传至 Botmaster 的能力.

D-BitBot 加入僵尸网络后连接至比特币主网，监听所有被广播的交易，从中标识出由 Botmaster 发

出的交易并解析对应的命令. Botmaster 可通过命令发布信道将 Normal Bot 升级为 Sensor Bot,加强对僵尸网络的控制. D-BitBot 的通信细节将在 C&C 通信机制中进行介绍. D-BitBot 的架构如图 1 所示.

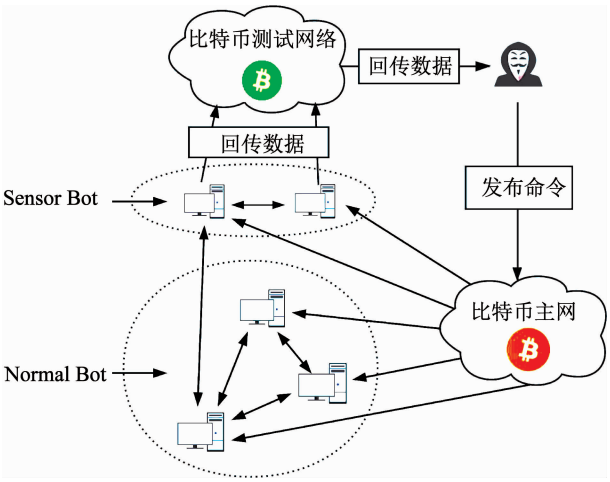


**图 1 D-BitBot 的架构**

Fig. 1 Architecture of the D-BitBot

## 2 D-BitBot 的 C&C 通信机制

### 2.1 通信数据载体

利用 OP_RETURN 输出脚本函数[23],任何数据均可被嵌入到比特币交易中. 这是 Bitcoin Core 0.9.0版本发布的新函数. 比特币用户可在单次交易中嵌入最多 83 字节的字节数据. 这个数据段长度足以让 Botmaster 对僵尸网络进行命令发布.

### 2.2 D-BitBot 的通信信道

#### 2.2.1 命令发布信道

Botmaster 通过合法的比特币交易进行命令发布. Botmaster 拥有一个公私钥对. 私钥可用于发送比特币交易,公钥被硬编码在 D-BitBot 中,用于校验数字签名,标识来自 Botmaster 的交易. Botmaster 将命令数据嵌入比特币交易中,用私钥签名后发送到比特币主网中. D-BitBot 连接至比特币网络后,标识来自 Botmaster 的交易以接收命令. 命令发布和接收的过程如图 2 所示.

#### 2.2.2 上行通信信道

每个 Sensor Bot 拥有两个由 Botmaster 分配的比特币测试网络公私钥对以及对应的两个比特币测试网络地址,Sensor Bot 创建比特币交易时,将这两个地址分别作为交易的输入和输出地址,然后将回传数据使用 OP_RETURN 脚本函数嵌入该交易中,再通过比特币测试网络将交易发出. Botmaster 通过数字签名校验标识来自 Sensor Bot 的交易并接收回传数据. 上行通信过程如图 3 所示.

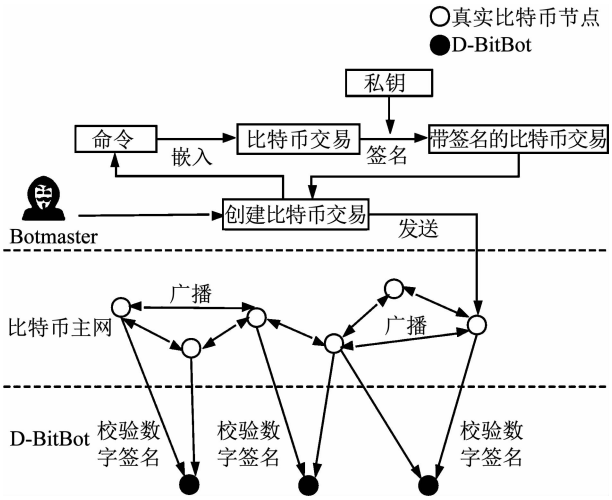


**图 2 命令发布过程**

Fig. 2 Command issuing process

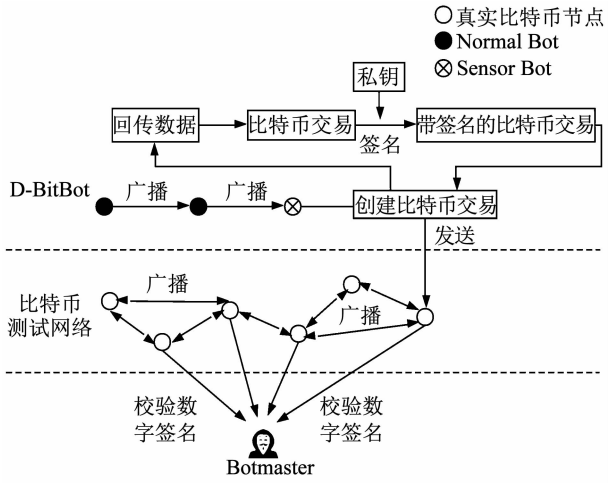


**图 3 上行通信过程**

Fig. 3 Upstream communication process

#### 2.2.3 D-BitBot 之间的通信信道

真实的比特币节点之间通过比特币消息[23]进行通信,如 version,verack,ping,tx 等. 每次进行通信之前,节点之间要先进行版本握手.

类似地,D-BitBot 在相互通信之前也要进行伪装的版本握手. 这样做可对通信双方的合法性进行校验,并实现流量混淆,使通信流量与真实比特币节点网络流量相似. D-BitBot 之间的通信数据被嵌在不可用的比特币交易中,实现流量伪装. D-BitBot 之间的通信过程如图 4 所示.

### 2.3 基于比特币区块链的上线方式

上线地址通过周期性发布的比特币交易被记录在比特币主网区块链上. 未加入网络的 D-BitBot 通过对比特币主网区块链进行检索即可获取最新的上线地址并加入到僵尸网络中. 即使安全研究人员分析出 Botmaster 使用的比特币地址,也难以通过屏蔽比特币地址的方式阻止上线地址的发布. 因为安全

研究人员无法使众多比特币矿工达成共识以屏蔽 Botmaster 所拥有的地址发出的比特币交易.

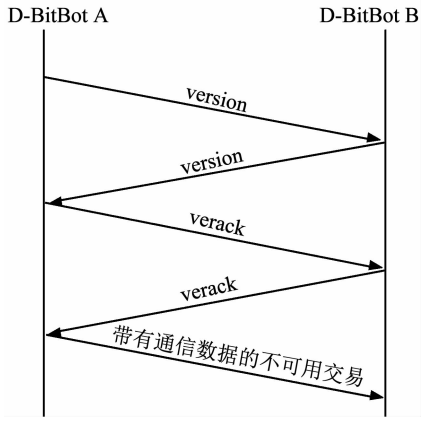


**图 4　D-BitBot 之间的通信过程**

Fig. 4　Process of communication among D-BitBots

### 2.4　基于 IP 地址加盐哈希排序的节点列表交换算法

对于节点列表请求算法，其设计目的是降低攻击者路由表节点注入的效率. 当所有在节点列表请求中收集到的节点信息被添加到请求节点的节点列表中的可能性相同，无论使用何种攻击方式，都不能高效地对该节点的路由表进行节点注入. 现有方法分析如下：

Sality：在 Sality 僵尸网络中，一个节点在收到节点请求时会随机选择节点列表中的一个节点进行回复. 针对 Sality，可通过多次请求以爬取其节点信息.

ZeroAccess：在 ZeroAccess 僵尸网络中，一个节点在收到节点请求时将选择一定数量的最近收到回复的节点进行回复. 该方式的一个严重的缺点是会泄露最近与该节点进行过通信的节点.

P2P Zeus：在 P2P Zeus 僵尸网络中，一个节点在收到节点请求时，该节点将对发出请求的节点到其节点列表中所有节点的异或距离进行排序，选出一定数量异或距离最短的节点进行回复. 针对该回复方式，可使用节点信息欺骗的方式改变当前节点到其他节点的异或距离，即可高效地爬取到单一节点列表中的所有节点.

为解决上述现有方法中存在的问题，抵抗路由表节点注入攻击和 P2P 僵尸网络节点爬取，本文提出一种基于 IP 地址加盐哈希排序的节点列表交换算法.

盐[28]在密码学中，是指在计算哈希值前将散列内容（例如：密码）的任意固定位置插入特定的字符串. 这个在散列中加入字符串的方式称为“加盐”. 其作用是让加盐后的散列结果和没有加盐的结果不相同，在不同的应用情景中，该处理可增加额外的安全性.

当一个节点 $Bot_{\mathrm{Req}}$ 需要节点填充自身的节点列表 $L_{\mathrm{Req}}$ 时，$Bot_{\mathrm{Req}}$ 首先计算所需节点数 $N_{\mathrm{need}}$，$M$ 为节点的容量. 若节点列表为空，$Bot_{\mathrm{Req}}$ 从比特币区块链上获取种子节点 $L_{\mathrm{seed}}$，否则向剩余节点发出节点请求. 收集所有回复到一个列表 $L_{\mathrm{temp}}$ 后，$Bot_{\mathrm{Req}}$ 将 $L_{\mathrm{temp}}$ 中的所有 IP 地址分别与 32 字节的随机字符串 Salt 结合，并计算其 SHA-256 哈希值，然后将结果存入 $L_{\mathrm{hash}}$ 中. 最后，$Bot_{\mathrm{Req}}$ 将对 $L_{\mathrm{hash}}$ 排序后对应的 IP 地址存入最终结果列表 $L_{\mathrm{Ret}}$ 中.

对于节点列表回复算法，一般来说，成功的节点爬取策略可获取足够的信息以绘制僵尸网络的网络地图，这是打击僵尸网络的重要一环. 由于 D-BitBot 在收到请求时返回的是节点列表的一个子集，因此攻击者需要对同一节点发出多次请求，才能获取更多节点. Karuppayah 等[27]提出基于特征值欺骗的节点爬取算法 ZeusMilker，以提高对同一节点的爬取效率. 为破坏相似的爬取策略，当一个节点 $Bot_{\mathrm{Rep}}$ 收到节点请求，$Bot_{\mathrm{Rep}}$ 首先生成一个 32 字节的随机字符串作为盐值 *Salt*，然后将请求节点的 IP 地址 $IP_{\mathrm{Req}}$ 和 *Salt* 结合并对其进行 SHA-256 哈希运算得到 $s_1$. 随后 $Bot_{\mathrm{Rep}}$ 对 $IP_{\mathrm{Req}}$ 进行 SHA-256 哈希运算得到 $s_2$，最后对 $s_1$ 和 $s_2$ 进行异或运算得到对 $IP_{\mathrm{Req}}$ 的特征值 $K_{\mathrm{Req}}$. 由于随机盐值不受攻击者操控，返回的节点列表将偏向于新的特征值 $K_{\mathrm{Req}}$，针对 D-BitBot 的特征值欺骗将变得低效. 最后 $Bot_{\mathrm{Rep}}$ 基于节点列表中节点到 $K_{\mathrm{Req}}$ 的异或距离选择 $S_{\mathrm{return}}$ 个节点进行回复.

算法 1 通过加入随机盐值保证节点添加的随机性，攻击者难以有效地对 D-BitBot 的节点列表进行节点注入攻击. 算法 2 通过加入随机盐值的方式生成针对特定 IP 地址的特征值，对于来自同一 IP 地址的多次请求，D-BitBot 将返回相同的结果. 对于使用 IP 地址进行特征值欺骗的节点爬取策略，加入随机盐值所生成新的特征值将使特征值欺骗变得低效. 算法 1 和算法 2 的实际效果将在节点列表交换算法分析实验中进行评估.

**算法 1**　D-BitBot 的节点列表请求算法

**输入：** $L_{\mathrm{Req}}$

**输出：** $L_{\mathrm{Ret}}$

1) $N_{\mathrm{need}} \leftarrow M - |L_{\mathrm{Req}}|$
2) if $N_{\mathrm{need}} == 0$ then
3)　　return $\varnothing$
4) $L_{\mathrm{Ret}} \leftarrow \varnothing$
5) if $|L_{\mathrm{Req}}| == 0$ then
6)　　$L_{\mathrm{seed}} \leftarrow$ GetSeed( )
7)　　for $i = 0; i < |L_{\mathrm{seed}}|; i++$ do
8)　　request($L_{\mathrm{seed}}[i]$)

9) else
10) for $i = 0; i < |L_{Req}|; i++$ do
11) request($L_{Req}[i]$)
12) $L_{temp} \leftarrow$GetResponse( )
13) $Salt \leftarrow$GenerateRandomSalt( )
14) if$L_{temp} \neq \varnothing$ then
15) for $i = 0; i < |L_{temp}|; i++$ do
16) $H_i \leftarrow$SHA $-$256Hash($L_{temp}[i] + Salt$)
17) $L_{hash} \leftarrow L_{hash} \cup \{(L_{temp}[i], H_i)\}$
18) Sort($L_{hash}$)
19) while $|L_{Ret}| < N_{need}$ && $|L_{hash}| > 0$ do
20) $L_{Ret} \leftarrow L_{Ret} \cup$GetIP($L_{hash}[0]$)
21) $L_{hash} \leftarrow L_{hash} - L_{hash}[0]$
22) return$L_{Ret}$
23) end

**算法 2** D-BitBot 的节点列表回复算法

**输入:** $L_{Rep}$, $IP_{Req}$

**输出:** $L_{Res}$

1) $L_{Res} \leftarrow \varnothing$
2) $Salt \leftarrow$GenerateRandomSalt( )
3) $s_1 \leftarrow$SHA $-$256Hash($IP_{Req} + Salt$)
4) $s_2 \leftarrow$SHA $-$256Hash($IP_{Req}$)
5) $K_{Req} \leftarrow$XOR($s_1$, $s_2$)
6) for $i = 0; i < S_{return}$ && $i < |L_{Rep}|; i++$ do
7) $L_{Res}[i] \leftarrow L_{Rep}[i]$
8) for $i = S_{return}; i < |L_{Rep}|; i++$ do
9) for $j = 0; j < S_{return}; j++$ do
10) $s_{temp1} \leftarrow$SHA $-$256Hash($L_{Rep}[i]$)
11) $s_{temp2} \leftarrow$SHA $-$256Hash($L_{Res}[j]$)
12) if XOR($s_{temp1}$, $K_{Req}$) < XOR($s_{temp2}$, $K_{Req}$) then
13) $L_{Res}[j] \leftarrow L_{Rep}[i]$
14) break
15) return $L_{Res}$
16) end

# 3 僵尸网络仿真分析

为对 D-BitBot 进行进一步分析,本文使用 PeerSim[29] 对 D-BitBot 进行 P2P 网络仿真分析.

## 3.1 僵尸网络的构建

根据相关研究,在文献[30-31]对僵尸网络的仿真中,僵尸网络的构建通过类似蠕虫传播的方式实现,因为在文献[30-31]提出的僵尸网络中,新节点不使用上线节点加入网络. 在本文的仿真中,僵尸网络中有一定数量的初始节点,上线地址从网络的所有节点中随机选出并周期性被更新. 新加入网络的 D-BitBot 向上线节点请求更多节点以填充自身的节点列表. 为方便分析,本实验随机生成的 IP 地址均为 IPv4 地址.

## 3.2 僵尸网络相关参数的定义

为更好地对僵尸网络进行分析,本文对僵尸网络的参数的定义在表 1 中列出.

**表 1 D-BitBot 相关参数的定义**

Tab. 1 Definition of parameters of D-BitBot

| 参数 | 描述 |
|---|---|
| $N$ | 僵尸网络当前总节点数 |
| $M$ | D-BitBot 的节点列表容量 |
| $p_{init}$ | Sensor Bot 在僵尸网络中的初始比重 |
| $p$ | Sensor Bot 在僵尸网络中的当前比重 |
| $T_{up}$ | 回传消息在僵尸网络中广播时的存活时间 |
| $N_c$ | 一个 D-BitBot 的节点列表中当前包含的节点个数 |
| $\mu_p$ | 所有 D-BitBot 的节点列表中平均包含的节点个数 |
| $N_s$ | 一个 D-BitBot 的节点列表中当前包含的 Sensor Bot 个数 |
| $\mu_s$ | 所有 D-BitBot 的节点列表中平均包含的 Sensor Bot 个数 |

## 3.3 僵尸网络的初始化

对于本文使用的僵尸网络初始化方式,每个 D-BitBot的节点列表近似于由整个僵尸网络中的节点随机填充. 因此,$N_s$的期望值为 $M \times p_{init}$. 根据相关研究[30-31],本文假设 $N$ 的初始值为 1 000,$M$ 的值为 20,$p_{init}$的值为 0.25,僵尸网络规模不断增长直至 $N$ 的值达到 20 000,在这个过程中 Normal Bot 不断地被升级为 Sensor Bot,使 $p$ 的值维持在 0.25 左右. 僵尸网络初始化完成后 $\mu_s$为

$$\mu_s = \frac{1}{N}\sum_{i=1}^{N} N_{s_i}. \tag{1}$$

式中 $N_{s_i}$为 D-BitBot i 节点列表中的 Sensor Bot 个数. $\mu_s$的实验值为 4.998,接近于 $N_s$的期望值 $M \times p_{init} = 5$. $N_s$的分布情况如图 5 所示.

由图 5 可知,近 80% 的 D-BitBot 节点列表中有 3 ~ 7 个 Sensor Bot. 该分布近似于平均值为 5 的正态分布. 为研究 $N_s$的离散度,本文对 $N_s$的标准差进行计算. 假设 $N_s$的标准差为 $\sigma$,则 $\sigma$ 为

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(N_{s_i} - \mu_s)^2}. \tag{2}$$

$\sigma$ 的实验值为 1.948,故 $N_s$具有一定的离散度,并可能会对 D-BitBot 的上线率产生影响. 本文将在鲁棒性评估实验中对上线率进行研究.

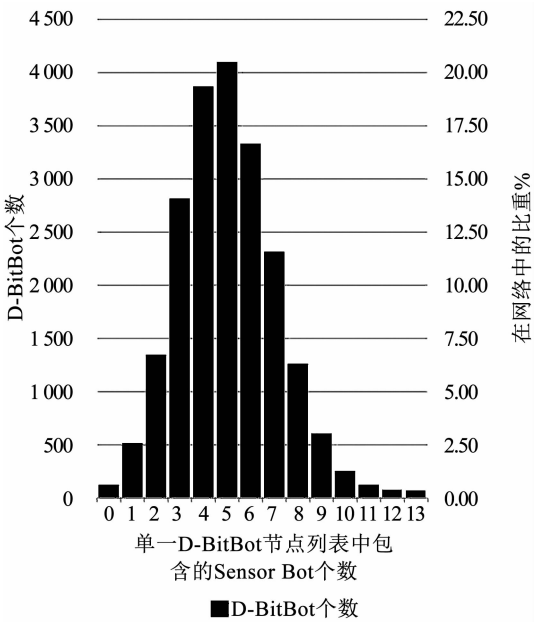


**图 5　单一节点列表中包含的 Sensor Bot 个数分布**

Fig. 5　Distribution of the current number of sensor bots in a single peer list

## 4　D-BitBot 鲁棒性评估

僵尸网络鲁棒性受多种因素影响，如回传信道失效、DDoS 攻击、女巫攻击、路由表投毒和僵尸节点离线等. 这些因素均会影响 D-BitBot 的上线率. 因此本文以D-BitBot的上线率评估 D-BitBot 的鲁棒性. 在本实验中，D-BitBot 的上线率为一个 D-BitBot 发出的消息能到达可用的 Sensor Bot 的概率. “可用”代表一个 Sensor Bot 没有被移除，能将数据通过回传信道发送到 Botmaster. 假设 $R(p_r)$ 为移除比重为 $p_r$ 的 Sensor Bot 后 D-BitBot 的上线率，则 $R(p_r)$ 的实验值为

$$R(p_r) = \frac{N_{\text{reachable}}}{N_{\text{remaining}}}. \tag{3}$$

式中 $N_{\text{reachable}}$ 为发出的消息能到达可用 Sensor Bot 的 D-BitBot 个数，$N_{\text{remaining}}$ 为僵尸网络中剩余的 D-BitBot 个数. 本文假设所有 Sensor Bot 在被移除前均为可用状态.

为计算 D-BitBot 发出的消息经过广播之后能够到达一个可用 Sensor Bot 的概率，需要先计算两个参数：$\mu_p$ 和 $p$. $\mu_p$ 为

$$\mu_p = \frac{1}{N}\sum_{i=1}^{N} N_{c_i}. \tag{4}$$

$p$ 随着 Sensor Bot 的移除而降低. $p$ 为

$$p = \frac{p_{\text{init}}(1-p_r)}{1-p_{\text{init}}\cdot p_r}. \tag{5}$$

接下来计算一条消息由 D-BitBot 发出之后到达一个可用的 Sensor Bot 的概率，即 $R(p_r)$ 的计算值. 事实上，只需计算出一条消息不能到达一个可用的 Sensor Bot 的概率，然后用 1 减去该概率即可得到 $R(p_r)$ 的计算值. 首先，一个 D-BitBot 不是可用的 Sensor Bot 的概率为 $(1-p)$，该 D-BitBot 节点列表中所有节点均不为可用的 Sensor Bot 的概率为 $(1-p)^{\mu_p}$，因此当 $T_{\text{up}}$ 为 1 时 $R(p_r)$ 的计算值为 $1-(1-p)\cdot(1-p)^{\mu_p}$. 以此类推可得 $R(p_r)$ 的计算值为

$$R(p_r) = 1-(1-p)^{\sum_{j=0}^{T_{\text{up}}}\mu_p}, \tag{6}$$

由式(6)可知，随着 $T_{\text{up}}$ 值的增大，由于指数函数的特性，$T_{\text{up}}$ 对 $R(p_r)$ 的值有决定性的影响. 当 $M$ 为 20，$p_{\text{init}}$ 为 0.25 时，随着 $T_{\text{up}}$ 的增长，$R(p_r)$ 接近于 1. 本章设计实验选出 $T_{\text{up}}$ 的最优值，避免网络负载过高且便于管理.

为通过网络仿真研究 $T_{\text{up}}$ 取不同的值时摘除 Sensor Bot 的比重 $p_r$ 对 D-BitBot 的上线率 $R(p_r)$ 的影响，针对自构建的仿真僵尸网络，本章使用不同摘除比重 $p_r$ 对网络中的 Sensor Bot 进行摘除，并对摘除后的网络连接率 $R(p_r)$ 进行监测，得到实验结果. $p_{\text{init}}$ 的取值为 0.25，$p_r$ 的取值范围为[0,1]，取值间隔为 0.05，$T_{\text{up}}$ 的取值范围为[1,4]，取值间隔为 1.

图 6 为在仿真环境中 $R(p_r)$ 计算值和实验值的比较. 随着 $T_{\text{up}}$ 的增长，$R(p_r)$ 的计算值趋近于 1. 当 $T_{\text{up}}\geqslant 2$，在 Sensor Bot 的移除比重 $p_r\leqslant 0.85$ 时，$R(p_r)$ 的实验值趋近于 1. 当 $T_{\text{up}}\geqslant 3$，在 Sensor Bot 的移除比重 $p_r=0.95$ 时，$R(p_r)\approx 0.9$. 这表明在 $T_{\text{up}}=3$ 时，僵尸网络具有良好的鲁棒性，因此当节点列表的最大值 $M$ 取值为 20，Sensor Bot 的初始比重 $p_{\text{init}}$ 取值为 0.25 时，$T_{\text{up}}$ 的最优取值为 3. $R(p_r)$ 的计算值与实验值基本相符，实验值略低于计算值. 这是因为在

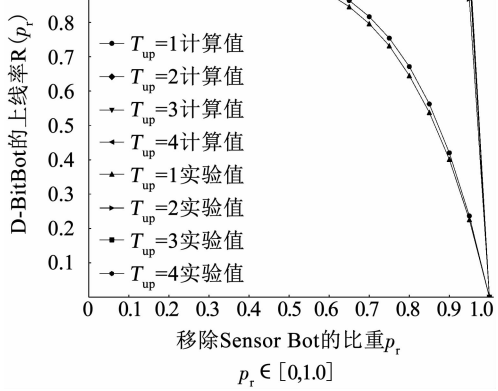


**图 6　$T_{\text{up}}$ 取不同的值时 $R(p_r)$ 的计算值和实验值的比较**

Fig. 6　Comparison of the calculated values and simulation results of $R(p_r)$ with different $T_{\text{up}}$

$R(p_r)$的计算式中使用的是$\mu_p$. 实际上,由自构建的仿真僵尸网络属性可得,单一 D-BitBot 节点列表中包含的 Sensor Bot 个数 $N_s$具有一定的离散程度,因此 $R(p_r)$的实验值和计算值存在误差. 虽然式(6)并不能计算出准确的 $R(p_r)$,但是,当安全研究人员没有获取实际数据时,式(6)仍可用于估算相似僵尸网络的上线率.

## 5 节点列表交换算法分析

### 5.1 节点注入抵御效果分析

为抵抗路由表节点注入攻击,降低节点注入的效率,算法 1 通过对 IP 地址加盐哈希后排序的方式实现对待添加节点的随机选取. 本章从自构建的仿真僵尸网络中随机选取一个节点列表中节点数为 $M$ 的节点,其中 $5/M$ 个节点被标记为恶意. 然后使其对节点列表中所有节点发起多次节点请求,然后计算每次经过算法 1 筛选后被添加的节点来自恶意节点的概率. 请求重复的次数为 30 000.

根据实验所得数据,经算法 1 筛选后被添加至节点列表中的节点来自恶意节点的平均概率为 19.97%,标准差为 7.8%. 证明经算法 1 筛选后所有候选节点能达到被随机选取添加至节点列表的效果.

### 5.2 节点爬取抵御效果分析

为降低攻击者对单一节点进行爬取的效率,本文提出基于异或距离的节点筛选算法,计算异或距离使用的特征值为请求节点的 IP 地址加盐后的 SHA-256 哈希值.

为评估该算法的效果,本文分别使用 ZeusMilker 节点爬取算法和随机生成特征值的爬取算法 Random 对本文方法、Sality、P2P Zeus 和 ZeroAccess 进行对比. 本文以节点发现率评估节点爬取的效果,即已发现的节点占该节点列表的节点总数的比重. 本章从自构建的仿真僵尸网络中随机选取 50 个节点,分别对不同的节点回复策略进行爬取,然后计算平均节点发现率,本文假设节点列表在被爬取的过程中不会发生变化,$M$ 的值为 20,$S_{\text{return}}$的值为 4. 图 7 和图 8 显示在不同的节点回复策略下 Random 算法和 ZeusMilker 算法对单一 D-BitBot 进行节点爬取的效率. 对于 ZeroAccess,不同的爬取方法其爬取结果相同,因为 ZeroAccess 固定返回节点列表中的前 $5/M$ 个节点.

由图 7 可知,除 ZeroAccess 外,随机爬取算法 Random 在经过大量的请求后均可获取到单一节点列表中的绝大部分节点. 本文提出的方法和 P2P Zeus 效果略优于 Sality.

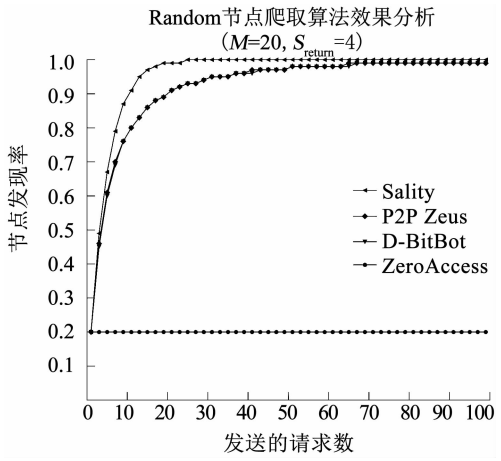


**图 7 Random 节点爬取算法对不同抵御方法的效果分析($M=20, S_{\text{return}}=4$)**

Fig. 7 Performance of Random on different anti-crawling strategies ($M=20$, $S_{\text{return}}=4$)

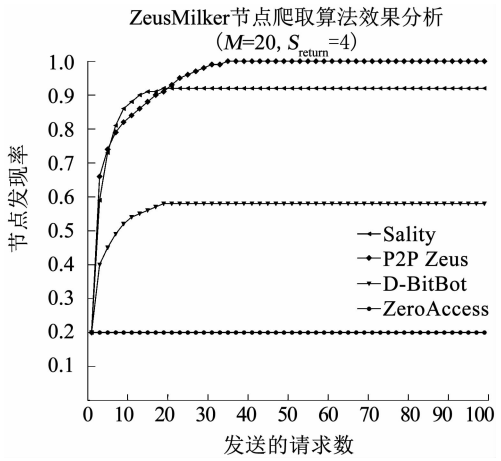


**图 8 ZeusMilker 节点爬取算法对不同抵御方法的效果分析($M=20, S_{\text{return}}=4$)**

Fig. 8 Performance of ZeusMilker on different anti-crawling strategies ($M=20$, $S_{\text{return}}=4$)

由图 8 可知,本文所提出的方法对 ZeusMilker 算法的抵御效果最佳,ZeusMilker 算法对其目标节点的平均发现率不足 60%. 对于 P2P Zeus,ZeusMilker 算法在一定数量的请求后的节点发现率可达 100%,而对于 Sality,ZeusMilker 算法最高节点发现率则略高于 90%.

实验结果证明本文提出的算法可有效抵御基于特征值欺骗的 ZeusMilker 节点爬取算法. 随机爬取算法 Random 无法通过少量请求获取单一 D-BitBot 的所有节点信息,存在效率低下的缺陷. 因此,本文提出提出的节点列表回复法可有效抵御僵尸网络节点爬取攻击.

## 6 相关讨论分析

### 6.1 抵御措施分析

#### 6.1.1 被感染端层面抵御措施

根据本文提出的僵尸网络模型设计方案及比特币协议，该模型需监听受感染主机的 8 333 和18 333 端口（比特币主网和比特币测试网络的默认通信端口）. 通过关闭受感染主机的对应端口可有效阻止其数据传输. 然而，关闭端口会对依赖该端口的其他网络服务产生影响，并会对真实比特币客户端的网络行为产生影响. 因此该操作仅作为应急措施而不能永久解除威胁.

6.1.2　比特币网络层面抵御措施

比特币网络并非由 Botmaster 所控制的节点组成的 P2P 网络，若属于 Botmaster 的比特币地址被网络中的节点所屏蔽或被列入黑名单，不仅会使本文提出的僵尸网络通信信道失效，还会使该僵尸网络控制端有被溯源的风险. 由于 96% 的比特币节点使用的客户端为 Bitcoin Core[32]，因此一种可能的抵御方式为安全研究人员与 Bitcoin Core 开发团队协商在程序中加入黑名单机制，屏蔽特定地址的交易或对发出交易的源 IP 地址进行追溯. 然而，黑名单机制违反比特币设计的理念和初衷[33]，会因比特币社区用户及开发人员的反对而无法实行.

6.1.3　网络服务提供商层面抵御措施

由 Botmaster 发出的交易数据须先经过被网络服务提供商（Internet Service Provider，ISP）发送到比特币网络，因此交易数据在被发出时可被部署于 ISP 层的基于软件定义网络（Software Defined Network，SDN）的针对特定协议的监测点[34]所捕获. 安全研究人员可根据捕获到的数据和信息进行进一步的溯源. 然而，比特币网络是一个节点分布在全球各地的 P2P 网络，安全研究人员难以与全球众多 ISP 协同部署对应的监测点.

### 6.2　相关讨论

6.2.1　经济成本分析

本文提出的实现方案通过比特币主网发送命令，每发送一条命令的成本约为 0.4 美元（0.1 m BTC），回传信道使用比特币测试网络进行数据回收，通信所需比特币可通过公开访问的网站免费获取，因此回传通信成本与比特币价格和僵尸网络的规模均无关. Sensor Bot 被捕获或被移除不会造成 Botmaster 的经济损失. 此外，节点间通信消息在自组建的 P2P 网络中广播，节点间无通信成本.

6.2.2　D-BitBot 信道分析

基于本文及相关研究所论述，对于比特币和以太坊等公开的区块链网络，暂无相关研究提出方法防止对其的恶意利用以及对恶意利用的有效溯源. 比特币主网和测试网络具有相同的功能和特性，两者均为拥有大量遍布全球的节点的 P2P 网络[35]，其区别仅在于交易数据的网络标识及协议使用的端口号不同，而且比特币网络没有独立的第三方组织对其进行监管，因此通过传统方法对比特币网络进行溯源并不可行.

本文提出的解决方案被控端和控制端在从比特币网络中接收消息时其网络行为与真实的比特币节点无异，被控端和控制端接收所有被广播的交易并在本地过滤出包含通信数据的交易. 使用比特币测试网络对通信数据进行回收，可利用比特币网络的可靠性和抗跟踪溯源性，同时降低回传信道的通信成本.

此外，该模型经少量修改即可部署至其他相似的公有区块链网络，将来可能会对网络空间安全造成威胁，应当引起安全研究人员的重视，并研究对应的防御技术.

## 7　结　论

僵尸网络已经成为当前网络空间面临的最大安全威胁之一. 针对基于公有区块链网络（如比特币、以太坊等）的僵尸网络通信模型研究中网络扩展代价高和回传通道容易被溯源的问题，本文提出 D-BitBot，一种基于比特币网络双向通信的 P2P 僵尸网络模型. 与其他对僵尸网络的相关研究相比，D-BitBot 的 C&C 信道更加难以被破坏和屏蔽，通信成本更低，且网络扩展性不受限于数字加密货币价格. 此外，本文提出基于比特币区块链的上线机制及基于 IP 地址加盐哈希排序的节点列表交换算法，使 D-BitBot 可避免单点故障的缺陷，并可有效抵御路由表节点注入攻击和僵尸网络节点爬取.

## 参考文献

[1] WERNER T. The miner botnet: Bitcoin mining goes peer-to-peer [EB/OL]. (2011-08-19) [2019-04-02]. https://securelist.com/the-miner-botnet-bitcoin-mining-goes-peer-to-peer-33/30863/

[2] GU Guofei, ZHANG Junjie, LEE Wenke. BotSniffer: Detecting botnet command and control channels in network traffic [C]// Proceedings of the 15th Annual Network and Distributed System Security Symposium. San Diego: The Internet Society, 2008

[3] STRAYER W T, LAPSELY D, WALSH R, et al. Botnet detection based on network behavior [J]. Botnet Detection, 2008: 1. DOI: 10.1007/978-0-387-68768-1_1

[4] ABU RAJAB M, ZARFOSS J, MONROSE F, et al. A multifaceted approach to understanding the botnet phenomenon [C]//Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement. Rio de Janeriro: ACM, 2006: 41. DOI: 10.1145/1177080.1177086

[5] NADJI Y, ANTONAKAKIS M, PERDISCI R, et al. Beheading hydras: Performing effective botnet takedowns [C]//Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. Berlin: ACM, 2013: 121. DOI: 10.1145/2508859.2516749

[6] STONE-GROSS B, COVA M, CAVALLARO L, et al. Your botnet is by botnet: Analysis of a botnet takeover[C]//Proceedings of the 16th ACM Conference on Computer and Communications Security. Chicago: ACM, 2009: 635-647. DOI: 10.1145/1653662.1653738

[7] KOVACS E. RAT abuses Yahoo mail for C&C communications[EB/OL]. (2014-08-04)[2019-04-02]. https://www.securityweek.com/rat-abuses-yahoo-mail-cc-communications

[8] NAPPA A, FATTORI A, BALDUZZI M, et al. Take a deep breath: A stealthy, resilient and cost-effective botnet using Skype[C]//Proceedings of Detection of Intrusions and Malware, and Vulnerability Assessment. Bonn: Springer, 2010: 81. DOI: 10.1007/978-3-642-14215-4_5

[9] PANTIC N, HUSAIN M I. Covert botnet command and control using Twitter[C]//Proceedings of the 31st Annual Computer Security Applications Conference. Los Angeles: ACM, 2015: 171. DOI:10.1145/2818000.2818047

[10] KATSUKI T. Malware targeting Windows 8 uses Google docs[EB/OL]. (2012-11-16)[2019-04-02]. https://www.symantec.com/connect/blogs/malware-targeting-windows-8-uses-google-docs

[11] HOLZ T, STEINER M, DAHL F, et al. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm[C]//Proceedings of First USENIX Workshop on Large-Scale Exploits and Emergent Threats. San Francisco: USENIX Association, 2008, 8(1): 1. DOI: 10.1109/MSECP.2003.1177002

[12] STARNBERGER G, KRÜGEL C, KIRDA E. Overbot: A botnet protocol based on Kademlia[C]//Proceedings of SecureComm. Turkey: ACM, 2008: 13. DOI:10.1145/1460877.1460894

[13] YAN Guanhua, CHEN Songqing, EIDENBENZ S. RatBot: Anti-enumeration peer-to-peer botnets[C]//Proceedings of Information Security, 14th International Conference. Xi´an: Springer, 2011. DOI: 10.1007/978-3-642-24861-0_10

[14] HERWIG S, HARVEY K, HUGHEY G, et al. Measurement and analysis of Hajime, a peer-to-peer IoT botnet[C]//Proceedings of Network and Distributed System Security Symposium. San Diego: The Internet Society, 2019. DOI: 10.14722/ndss.2019.23488

[15] ARCE I, LEVY E. An analysis of the Slapper worm[J]. IEEE Security Privacy, 2003, 1(1): 82. DOI: 10.1109/MSECP.2003.1177002

[16] FITZGIBBON N, WOOD M. Conficker. C: A technical analysis[EB/OL]. (2009-04-01)[2019-04-02]. https://www.sophos.com/fr-fr/medialibrary/PDFs/marketing% 20material/confickeranalysis.pdf

[17] TETARAVE S K, TRIPATHY S, KALAIMANNAN E, et al. eBot: Approach towards modeling an advanced P2P botnet[C]//Proceedings of 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). Rotorua: IEEE, 2018: 391. DOI: 10.1109/TrustCom/BigDataSE.2018.00065

[18] KARUPPAYAH S. Advanced monitoring in P2P botnets—A dual perspective[M]. 1st ed. Berlin, Heidelberg: Springer, 2018. DOI: 10.1007/978-981-10-9050-9

[19] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system[EB/OL]. (2008)[2019-04-02]. https://bitcoin.org/bitcoin.pdf

[20] WOOD G. Ethereum: A secure decentralised generalised transaction ledger[EB/OL]. (2014)[2019-04-02]. https://gavwood.com/paper.pdf

[21] ALI S T, MCCORRY P, Lee peter Hyuan-Jeen, et al. ZombieCoin: Powering next-generation botnets with bitcoin[C]//Proceedings of International Conference on Financial Cryptography and Data Security. San Juan: Springer, 2015: 34. DOI: 10.1007/978-3-662-48051-9_3

[22] ALI S T, MCCORRY P, LEE H J P, et al. ZombieCoin 2.0: Managing next-generation botnets using Bitcoin[J]. International Journal of Information Security, 2017. DOI: 10.1007/s10207-017-0379-8

[23] BINNS W. Bitcoin developer documentation[EB/OL]. (2019)[2019-04-02]. https://bitcoin.org/en/developer-documentation

[24] PIROZZI A. BOTCHAIN aka The dark side of blockchain[EB/OL]. (2018-10-25)[2019-04-02]. https://www.slideshare.net/AntonioPirozzi4/botchain-aka-the-dark-side-of-blockchain

[25] MALAIKA M. Botract—Abusing smart contracts and blockchain for botnet command and control[EB/OL]. (2017-11-15)[2019-04-02]. https://sector.ca/sessions/botract-abusing-smart-contracts-and-blockchain-for-botnet-command-and-control/

[26] CHINCHILLA C. Ethereum development tutorial[EB/OL]. (2019-03-16)[2019-04-02]. https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial

[27] KARUPPAYAH S, ROOS S, ROSSOW C, et al. Zeus Milker: Circumventing the P2P Zeus neighbor list restriction mechanism[C]//Proceedings of 2015 IEEE 35th International Conference on Distributed Computing Systems. Columbus: IEEE Computer Society, 2015: 619. DOI: 10.1109/ICDCS.2015.69

[28] BOKLAN K D. Large key sizes and the security of password-based cryptography[J]. International Journal of Information Security and Privacy(IJISP), 2009, 3(1): 65. DOI:10.4018/jisp.2009010105

[29] MONTRESOR A, JELASITY M. PeerSim: A scalable P2P simulator[C]//Proceedings of 2009 IEEE Ninth International Conference on Peer-to-Peer Computing. Seattle, Washington: IEEE, 2009: 99-100. DOI: 10.1109/P2P.2009.5284506

[30] WANG Ping, SPARKS S, ZOU C C. An advanced hybrid peer-to-peer botnet[J]. IEEE Transactions on Dependable and Secure Computing, 2010, 7(2): 113. DOI: 10.1109/TDSC.2008.35

[31] LIU Chaoge, LU Weiqing, ZHANG Zhiqi, et al. A recoverable hybrid C&C botnet[C]//Proceedings of 6th International Conference on Malicious and Unwanted Software. Fajardo: IEEE Computer Society, 2011: 110. DOI: 10.1109/MALWARE.2011.6112334

[32] COINDANCE. Coin dance: Bitcoin nodes summary[EB/OL]. (2019)[2019-04-02]. https://coin.dance/nodes

[33] BUSTILLOS M. The Bitcoin boom[EB/OL]. (2013-04-01)[2019-04-02]. https://www.newyorker.com/ tech/annals-of-technology/the-bitcoin-boom

[34] HAQ O, ABAID Z, BHATTI N, et al. SDN-inspired, real-time botnet detection and flow-blocking at ISP and enterprise-level[C]//Proceedings of 2015 IEEE International Conference on Communications (ICC). London: IEEE, 2015: 5278. DOI: 10.1109/ICC.2015.7249162

[35] BITNODES. Bitnodes: Global Bitcoin nodes distribution[EB/OL]. (2019)[2019-04-02]. https://bitnodes.earn.com/

(编辑 苗秀芝)