doi:10.3969/j.issn.1000-436x.2014.01.018

基于覆盖率分析的僵尸网络控制命令发掘方法

王志，蔡亚运，刘露，贾春福
（南开大学 计算机与控制工程学院，天津 300071）

摘 要：从僵尸程序执行轨迹对二进制代码块的覆盖规律出发，提出了一种僵尸网络控制命令发掘方法。通过分析执行轨迹对代码块的覆盖率特征实现对僵尸网络控制命令空间的发掘，根据代码空间是否被全覆盖来验证发现的僵尸网络命令空间的全面性。对僵尸网络 Zeus、SdBot、AgoBot 的执行轨迹进行了代码块覆盖率分析，结果表明，该方法能够快速准确地发掘出僵尸网络的控制命令集合，时间和空间开销小，且该命令集合所对应的执行轨迹可以覆盖僵尸程序 95%以上的代码空间。

关键词：恶意代码分析；僵尸网络；命令与控制协议；基本块；覆盖率
中图分类号：TP393.08 **文献标识码**：A **文章编号**：1000-436X(2014)01-0156-11

Using coverage analysis to extract Botnet command-and-control protocol

WANG Zhi, CAI Ya-yun, LIU Lu, JIA Chun-fu
(College of Computer and Control Engineering, Nankai University, Tianjin 300071, China)

Abstract: There are some inherent patterns in the bot execution trace coverage of basic blocks. Using these patterns, an approach was proposed to infer Botnet command-and-control protocol (C&C protocol). Without intermediate representation of binary code and constraints solving, this approach has a lower time and space overhead. This coverage analysis approach was evaluated on 3 famous Botnet: Zeus, Sdbot and Agobot. The result shows that this approach can accurately and efficiently extract the Botnet control commands. And the completeness of the extracted control commands could be verified by checking whether all available basic blocks in bot are covered by the traces triggered by the control commands.

Key words: malware analysis; Botnet; command-and-control protocol; code block; code coverage

1 引言

僵尸网络[1~4]和其他恶意软件（如蠕虫）的一个重要不同之处是僵尸网络由命令与控制协议（C&C protocol）驱动，即僵尸网络的控制者（bot master）通过远程控制命令与多台受控主机（bot）进行通信。这种控制机制使僵尸网络具有灵活性、私密性、可控性，使攻击者能够以极低的代价获得强大的分布式计算能力和丰富的信息资源，例如发动分布式拒绝服务攻击[5]、发送大量的垃圾邮件[6]、分发间谍程序和广告程序、窃取僵尸主机的用户信息等。僵尸网络是当前网络安全的最大威胁，而且在不断地演化和变异，并已经渗透到移动互联网、云计算平台、甚至工业控制系统中，如图 1 所示，其危害已经影响到国家安全，甚至到军事安全等多个重要领域。

对僵尸网络的研究可以归纳为 5 个方面：检测[7~16]、追踪[17~23]、测量[24~26]、预测[27~33]和对抗[34~36]，

收稿日期：2013-03-14；修回日期：2013-10-22
基金项目：国家自然科学基金资助项目(61300242，61272423，60973141)；国家重点基础研究发展计划(“973”计划)基金资助项目(2013CB834204)；中央高校基本科研业务费专项基金资助项目(65121012)；南开大学—腾讯联合基金资助项目(2011-11)
Foundation Items: The National Natural Science Foundation of China(61300242, 61272423, 60973141); The National Basic Research Program of China(973 Program)(2013CB834204); The Fundamental Research Funds for the Central Universities (65121012); Nankai University-Tencent Joint Project (2011-11)

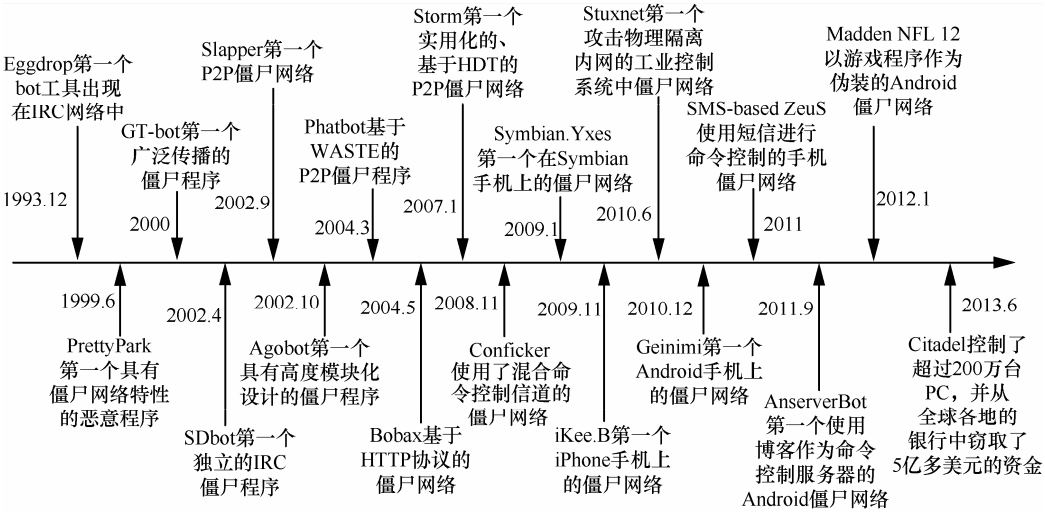


图 1　僵尸网络的演化进程

如图 2 所示。僵尸网络自身也在不断地进行变异，逐渐增强其自我保护能力[7,16,37]。因此，防治僵尸网络的唯一办法就是连续不断地跟踪僵尸网络技术的新进展，探索僵尸网络所固有的脆弱性，研究更加高效、准确、全面的僵尸网络分析和防治技术以应对新的网络安全威胁。

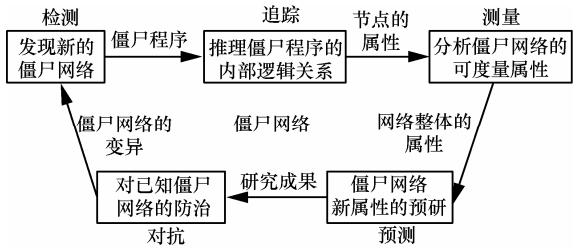


图 2　僵尸网络的研究内容

僵尸网络的脆弱性之一就是僵尸程序执行过程中的路径信息泄露问题。泄露的路径信息是僵尸网络内部命令控制逻辑关系的二进制表示，利用动态污点分析、符号执行、约束求解等技术可以对这些路径信息进行收集、形式化表示和推理，实现逆向构建僵尸网络的命令控制模型[19]。但是，二进制执行轨迹中缺乏高级语言里的类型信息、结构信息等，而且 CPU、操作系统等的控制逻辑都混杂在执行轨迹的路径信息中，导致路径信息的形式化推理过程复杂、耗时，并受到路径空间爆炸问题的制约，难以保证路径空间遍历的全面性。

与执行轨迹泄露的路径信息一样，僵尸程序不同的执行轨迹对二进制代码块的覆盖信息也是易于收集，本文通过挖掘这些覆盖信息的内在规律，提出了一种二进制代码覆盖率分析算法，不需要对二进制代码进行中间表示与约束求解，在较小的时间开销和空间开销下，实现从僵尸程序的执行轨迹中快速发掘僵尸网络的控制命令集合。在不考虑 ROP[38]和代码移动[39]的情况下，该方法能够通过计算有效代码空间是否被全覆盖来验证发现的僵尸网络命令集合是否全面。

2　相关工作

目前，僵尸网络命令与控制协议的分析方法主要有 2 类：基于流量的分析和基于二进制代码的分析。

基于流量的分析方法有 CUI 等人[40]提出的通过统计网络数据流中不同协议字段的出现次数来推断协议结构的方法等。

基于二进制代码的分析利用 IDAPro，W32Dasm 等工具对二进制代码进行静态反汇编，Vigna[41]提出了利用反汇编方法静态地提取二进制代码中包含的信息来还原高级语言的语义。CABALLERO 等人[42]提出通过对缓冲区解析自动逆向分析僵尸网络通信协议的方法。CHO 等人[23]提出了使用协议状态机分析僵尸网络命令与控制协议的方法。这些方法主要集中在僵尸网络通信协议的格式上，分析僵尸程序发送和接收信息的语法格式及每个域的语义。这些方法能够实现协议字段的自动划分和具有特殊协议字段的识别，但是缺乏对控制命令的全面性和完整性的证明。LIM 等人[43]提出一种基于系统 API 调用和控制依赖图的提取控制命令的方法。该方法静态地从僵尸程序可执行文件中提取触发 API 级行为的命令字符串及其和 API 调用实参的关

系。该方法可以有效地提取出部分僵尸程序的控制命令，但对系统 API 调用有较强的依赖性，且不能处理经压缩的或加密的僵尸程序。王志等人[44]利用恶意代码的路径信息泄露问题提出了推理恶意代码的主机环境依赖性的方法，并通过二进制代码插装技术削弱了恶意代码对主机环境的感知能力，但是该方法没有对网络环境的依赖性进行分析，不能推理僵尸程序的恶意行为与网络输入数据之间的依赖关系。

3 基于覆盖率分析的僵尸网络控制命令发掘方法

僵尸程序的路径空间具有复杂性，直接进行路径空间的遍历会遇到路径空间爆炸问题。本文利用动态执行轨迹对代码块的覆盖率将静态的代码空间与动态的路径空间建立了联系，提出了一种基于统计特征分析的僵尸网络控制命令发掘方法，利用僵尸程序执行轨迹对二进制代码块的覆盖信息的内在规律，从有限的执行轨迹集合中，实现对僵尸网络控制命令空间的快速发掘。基于覆盖率的分析方法减少了对路径空间遍历的依赖性，时间和空间开销较小，更具有实用性。

3.1 执行轨迹的基本块覆盖

3.1.1 执行轨迹与路径空间

执行轨迹是僵尸程序路径空间中一条路径的执行过程的详细记录，其中记录了执行过程中 CPU 所执行过的每一条指令、指令执行前后的 CPU 标志位状态、与指令相关的寄存器和内存等。执行轨迹中虽然缺乏高级语言源代码中的结构信息、类型信息等，但是它详细记录了高级语言中各种算法、数据结构、逻辑关系在 CPU 上用二进制机器代码的具体实现过程，是程序语义的二进制表示，其中就包含了程序内在的控制逻辑关系。

执行轨迹的获取通常是使用虚拟机技术或二进制代码插装技术实现，例如基于 QEMU[45]虚拟机的 BitBlaze[46]平台和 Intel 公司提供的二进制代码插装工具 Pin[47]。目前，CPU 的主频已经达到 GHz 级别，每秒钟 CPU 执行的指令数量将是千万级的，因此一条详细的程序执行轨迹的体积非常庞大。通常，执行轨迹的捕获过程会使用动态污点传播技术，只记录部分指令的执行过程，从而减小执行轨迹的体积。

执行轨迹记录了程序一条路径的执行过程。程序所有可达的路径集合构成了这个程序的路径空间。二进制代码的路径空间可以用一个二叉树表示，根节点就是程序的入口点（entry point），路径分支是由执行轨迹中的 CPU 条件跳转指令通过判断跳转条件是否满足而产生。执行轨迹是路径空间这棵二叉树从根节点到一个叶子节点的路径。路径空间与控制流图的区别是路径空间中没有循环结构，而控制流图中有循环结构。由于程序中有些循环结构的退出条件的不确定性，导致路径空间爆炸问题，从而限制了通过遍历路径空间实现对程序内部控制逻辑关系的挖掘。对路径空间遍历时通常会采用一些启发式的方法，例如离线（offline）分析中限制循环的展开次数，或者在线（online）分析中限制循环的执行时间等。

3.1.2 基本块（basic block）与代码空间

基本块是一段顺序执行的 CPU 指令序列，其中只有一个入口和一个出口。基本块的第一条指令被执行了，后面的指令会顺序执行，直到最后一条指令，中间没有控制流的跳转。执行轨迹中的 CPU 指令数量非常大，在控制逻辑的分析中，一般不以 CPU 指令为最小单位，而是以基本块为最小单位，路径空间中二叉树的每一个节点都是一个基本块。

程序的代码空间是程序中所有 CPU 指令的集合。利用基本块的定义可以对程序的代码空间进行划分，进而用基本块的集合表示代码空间。代码空间与路径空间都是程序的一种表示方式，代码空间是有限的，遍历代码空间的开销较低，而路径空间由于路径爆炸问题导致遍历路径空间的开销非常大。本文一个研究内容就是在一定的前提条件下，利用遍历代码空间模拟对路径空间的遍历，实现对僵尸网络控制命令集合完整性和全面性的验证。

3.1.3 执行轨迹的基本块覆盖

路径空间是对僵尸程序动态执行过程的描述，而代码空间是对僵尸程序静态代码分布的描述。本文通过路径空间的执行轨迹对代码空间的基本块覆盖信息，将僵尸程序的路径空间与代码空间建立联系，如图 3 所示。

路径空间是用二叉树表示的，描述了程序的动态执行过程，从根节点到一个叶子节点的路径就是一条执行轨迹。代码空间是用有限个基本块组成的集合表示的，其中不包含执行时的先后顺序与执行次数，只是对代码所在内存空间的一个划分。

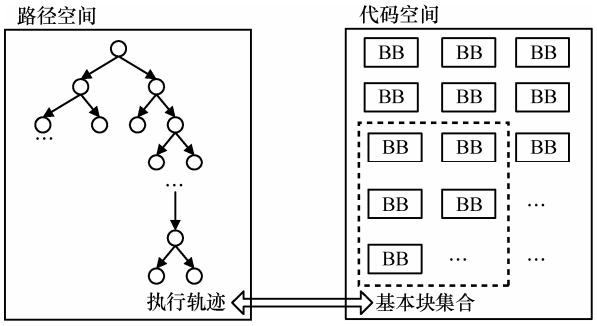


图 3　路径空间的执行轨迹在代码空间覆盖的基本块集合

如果在程序的执行过程中，某些基本块被执行，则称这些被执行的基本块被执行轨迹覆盖。被一条执行轨迹覆盖的基本块是代码空间的一个子集。基本块的覆盖信息是指被执行路径覆盖的所有基本块的执行信息，其中包括被覆盖基本块的执行顺序与执行次数。

3.2　基本块的覆盖率与覆盖次数

基本块被执行轨迹覆盖的信息包括两部分：覆盖率和覆盖次数。覆盖率 P_{BB} 描述的是一个基本块被执行轨迹覆盖的频繁程度 $P_{\mathrm{BB}}=\dfrac{N_{\mathrm{cover}}}{N_{\mathrm{all}}}$，$N_{\mathrm{all}}$ 是当前已捕获的执行轨迹的总数，N_{cover} 是覆盖该基本块的执行轨迹数量。覆盖次数 V_{BB} 描述的是基本块在不同执行轨迹中被执行的频繁程度，它是一个向量 $V_{\mathrm{BB}}=(V_{T_1},V_{T_2},\cdots,V_{T_n})$，$V_{T_1},V_{T_2},\cdots,V_{T_n}$ 表示基本块 BB 在执行轨迹 $T_1,T_2,\cdots,T_n$ 中被覆盖的次数。

根据基本块 BB 在已捕获的执行轨迹中的覆盖率 P_{BB} 和覆盖次数 V_{BB}，可以将基本块分成 5 类，如表 1 所示。

表 1　基于覆盖率和覆盖次数的基本块分类规则

覆盖率	覆盖次数	分类
P_{BB}=100%	具有相似性	1
	不具有相似性	2
0< P_{BB} <100%	不具有唯一性	3
	具有唯一性	4
P_{BB} =0	(0, 0,⋯, 0)	5

根据基本块是否被所有已捕获的执行轨迹覆盖，把基本块分成 2 类：第一类是被当前轨迹都覆盖到的基本块，P_{BB}=100%；第二类是只被部分执行轨迹覆盖到的基本块，P_{BB}<100%。P_{BB}=100%的基本块根据其在不同执行轨迹中的覆盖次数是否具有相似性可以分成 2 类：类型 1 和类型 2。类型 1 的基本块是 P_{BB}=100%，且在某 2 个执行轨迹中被执行的次数具有相似性，即 $\exists T_i,T_j$，$V_{T_i}==V_{T_j}$；类型 2 的基本块是 P_{BB}=100%，且基本块在不同的执行轨迹中的执行次数没有相似性，即 $\forall T_i,T_j$，$V_{T_i}!=V_{T_j}$。P_{BB}<100%的基本块根据其是否只被唯一的一条执行轨迹覆盖也可以分成 2 类：类型 3 和类型 4。类型 3 的基本块是 P_{BB}<100%，且存在至少 2 条执行轨迹 T_i 和 T_j，$V_{T_i}!=0$ 且 $V_{T_j}!=0$；类型 4 的基本块是 P_{BB}< 100%，且只有唯一的一条执行轨迹执行了该基本块，在其他执行轨迹中的执行次数都是 0。

第 5 类基本块是指那些没有被任何执行轨迹覆盖过的代码块，P_{BB}=0。

3.3　覆盖率分析算法

僵尸程序的执行轨迹是其程序语义在 CPU 上的二进制表示，其中包含了僵尸网络的命令控制逻辑。本节将详细介绍如何利用基本块的覆盖率分析从僵尸程序的执行轨迹中发掘出其所在僵尸网络的控制命令集合。

3.3.1　命令控制逻辑所在基本块的特征分析

僵尸程序的命令控制逻辑的实现方式主要有循环结构和分支结构 2 类，伪代码如下所示。

基于循环结构的命令控制逻辑的伪代码为

```
While
{
    If (input==cmd)
    {
        Trigger related behaviors
    }
    cmd=next commanding command set
}
```

基于 if-else 分支结构的命令控制逻辑的伪代码为

```
if (input==cmd_1)
{
    trigger related behaviors
}
else if (input== cmd_2)
{
trigger related behaviors
}
…
else if (input == cmd_n)
```

```
{
    trigger related behaviors
}
…
```

这些命令控制逻辑会分散到几个不同的基本块中。僵尸程序的一条执行轨迹会覆盖上百万个基本块，即使去掉了操作系统空间、动态链接库 DLL 等非来自于僵尸程序所在文件的基本块后，执行轨迹中还会有成千上万的基本块。如何从大量的基本块中发掘出包含僵尸程序命令控制逻辑的基本块是一个挑战。

将僵尸程序基本块的功能主要分成了 3 类：预处理、命令控制逻辑与攻击行为。预处理功能主要包括代码和数据的加密解密、接入僵尸网络、控制命令的侦听、Rootkit 与注入等自我隐藏功能，多态、变型等躲避杀毒软件的功能等；命令控制逻辑就是要挖掘的对象，它负责对僵尸网络控制命令的解析，并触发命令所对应的攻击行为；攻击行为是指由控制命令所触发各种恶意行为，例如，DDoS 攻击、发送垃圾邮件、盗取用户数据等。

3.3.2 基于循环结构的命令控制逻辑基本块的特征分析

本文分析命令控制逻辑功能的基本块与其他 2 种功能的基本块在执行轨迹覆盖率和覆盖次数上的差异，实现对僵尸网络控制命令的挖掘。首先，从执行轨迹的基本块覆盖率上，预处理功能的基本块和命令控制逻辑的基本块在每条执行轨迹上都会被执行，因此它们的覆盖率 P_{BB}=100%，而攻击行为的基本块由于不同命令所触发的攻击行为不同，所以某一攻击行为的基本块不会出现在所有的执行轨迹中，所以其覆盖率 P_{BB}<100%；其次，从覆盖次数上分析，预处理功能的基本块在僵尸程序每次运行时的执行过程具有相似性，因此在某 2 个执行轨迹中被执行的次数会具有相似的情况，即 $\exists T_i, T_j$，$V_{T_i} == V_{T_j}$。命令控制逻辑所在的基本块虽然每条执行轨迹都会执行，但是对于不同的控制命令控制逻辑基本块的执行次数是具有差异性的。通过分析可以得到 2 条识别命令控制逻辑基本块的规则。

规则 1 $P_{\mathrm{BB}} = 100\%$

规则 2 $\forall T_i, T_j$，$V_{T_i} != V_{T_j}$

通过规则 1 和规则 2 可以过滤掉大量的非控制逻辑基本块。从 Zeus、Agobot 和 Sdbot 的实验结果发现，虽然有大量的代码块被过滤，符合规则 1 和规则 2 的代码块仍然很多，因此需要继续去除噪音的方法。提出了 2 种去除噪音的策略：策略 1 是增加执行轨迹的数量，从而增强规则 1 和规则 2 的过滤效果。增加执行轨迹的数量可以发现轨迹间覆盖率的更多差异，从而减少 P_{BB}=100%的基本块数量，增加规则 1 的过滤强度；增加执行轨迹的数量也会使覆盖次数向量 $V_{\mathrm{BB}} = (V_{T_1}, V_{T_2}, \cdots, V_{T_n})$ 变长，更容易发现相似的覆盖次数，从而增加规则 2 的过滤强度；策略 2 是不增加执行轨迹的数量，通过创建噪音过滤规则实现对基本块的二次过滤，由于僵尸网络的控制命令数量并不是很多，因此命令控制逻辑基本块的覆盖次数的上限是比较少的。设置一个阈值 *th*，覆盖次数超过 *th* 的基本块就认为是噪音；又对命令控制逻辑基本块的在执行轨迹里的执行顺序进行分析，僵尸程序在进行完逻辑判断后通常会立即响应这些命令，即触发响应的攻击行为，因此命令控制逻辑基本块后面跟着的应该是攻击行为基本块，只有唯一的一条执行轨迹覆盖该基本块。噪音的过滤规则如下。

规则 3 $\forall T_i$，$V_{T_i} < th$

规则 4 BB_i 满足规则 1、规则 2、规则 3，且 $P_{\mathrm{BB}_{i+1}} = \dfrac{1}{N_{\mathrm{all}}}$

识别循环结构命令控制逻辑代码块的伪代码如下。

算法 1 循环结构命令控制逻辑基本块识别算法

输入：*TRACES* 僵尸程序执行轨迹集合;
th 覆盖次数的阈值

输出：*SET_BB* 僵尸程序命令控制逻辑基本块集合

```
Searching(TRACES, th){
SET_BB={}; //基本块集合
SET_P_BB={};//基本块覆盖率集合
SET_V_BB={};//基本块覆盖次数集合
foreach(t in TRACES){
(SET_BB, SET_P_BB, SET_V_BB)=update_coverage(t);
  //更新基本块的覆盖信息
}
  // 根据 4 个规则过滤非命令控制逻辑的基本块
  SET_BB = analyzing with the RULES (SET_
```

BB, *SET_P*$_{BB}$, *SET_V*$_{BB}$, *th*);

return *SET_BB*;

}

算法的输入是执行轨迹集合 *TRACES* 和覆盖次数的阈值 *th*，输出是找到的包含僵尸网络命令控制逻辑的基本块。算法的执行过程主要分为两部分：收集执行轨迹集合 *TRACES* 中的基本块覆盖信息；根据 4 个已定义的规则对非命令控制基本块进行过滤。

3.3.3 基于分支结构的命令控制逻辑基本块的特征分析

分支结构的命令控制逻辑基本块的覆盖信息特征不同于循环结构的覆盖信息特征，其不同点主要在以下 2 个方面：首先，基本块的覆盖率不一致，基于循环结构的基本块在不同的执行轨迹中都会被覆盖 P_{BB}=100%，但是基于 if-else 分支结构的基本块并不是全部被 100%覆盖。因为僵尸程序对控制命令的理解过程总是从 if 开始，这样 if 所在的基本块的覆盖率 P_{if} =100%，而且覆盖次数 V_{if}=(1, 1,⋯,1)，每个执行轨迹只执行 1 次，因此 if 所在的基本块是第一类基本块；然后，如果 if 所在基本块不能告诉僵尸程序如何理解控制命令，那么按顺序一个 else 接着一个 else 的对控制命令进行理解。当知道了如何去响应这个命令后，僵尸程序就不会再继续执行后面的 else 代码了。因此，else 所在的基本块的覆盖率 P_{else}<100%，V_{else}=(1 or 0, 1 or 0,⋯,1 or 0)，而且不同 else 基本块的总覆盖次数 $\sum_{i=1}^{n} V_{T_i}$（n 是捕获的执行轨迹数量）也是不一致的。因此可以总结出如下识别规则。

规则 1　$P_{if}=100\%$，且 $V_{if}=(1,1,\cdots,1)$

规则 2　$P_{else}<100\%$，

且 $V_{else}=(1 \text{ or } 0,1 \text{ or } 0,\cdots,1 \text{ or } 0)$

规则 3　$V_{else}=\sum_{i=1}^{n} V_{T_i}$，

且 $\forall BB_{else\_i}, BB_{else\_j}$，$V_{esle\_i} != V_{esle\_j}$

在执行轨迹中，if 所在的基本块后面会跟着若干个 else 基本块。但是，僵尸程序中基于分支结构的命令控制逻辑并不像高级语言源代码中的 if-else 结构那样非常规范。在真实的僵尸程序执行轨迹里面，分支结构的命令控制逻辑基本块中会夹杂着很多噪音，例如预处理功能基本块与命令控制基本块混杂在一起等。因此，要得到清晰的命令控制逻辑的分支结构，需要对噪音基本块进行过滤。这些夹杂的噪音一般是属于第 4 类基本块，也就是 $P_{BB}=\frac{1}{N_{all}}$。去掉噪音基本块后，if 基本块后面的 else 基本块的数量也是有其内在规律性的，即不同的执行轨迹中 if 后面的 else 基本块的数量是不一致的，即 $\forall T_i, T_j$，*Num*(BB$_{else}$ in T_i)!=*Num*(BB$_{else}$ in T_j)。因此可以总结出如下识别规则。

规则 4　删除 $P_{BB}=\frac{1}{N_{all}}$ 的基本块

规则 5　$\forall T_i, T_j$，

Num(BB$_{else}$ in T_i)!=*Num*(BB$_{else}$ in T_j)

识别分支结构命令控制逻辑代码块的伪代码如下。

算法 2　分支结构命令控制逻辑基本块识别算法

输入：***TRACES*** 僵尸程序执行轨迹集合;

输出：***SET_BB*** 僵尸程序命令控制逻辑 if 基本块

Searching(*TRACES*){

*SET*_BB={}; //基本块集合

SET_P$_{BB}$={};//基本块覆盖率集合

SET_V$_{BB}$={};//基本块覆盖次数集合

foreach (*t* in *TRACES*){

//更新基本块的覆盖信息

(*SET_BB*, *SET_P*$_{BB}$, *SET_V*$_{BB}$) = *update_ coverage*(*t*);

}

// 根据 5 个规则定位 if 基本块的位置

SET_BB = analyzing with the *RULES* (*SET_BB*, *SET_P*$_{BB}$, *SET_V*$_{BB}$);

return *SET_BB*;

}

4　实验结果与分析

为了验证基于覆盖率分析的僵尸网络控制命令发掘方法的有效性，在 Zeus、Agobot 和 Sdbot 僵尸网络的样本上进行了实验分析。实验环境为 Intel Core2 Quad Q9400 2.66 GHz 处理器、4 GB 内存、Ubuntu 12.04 操作系统。

针对僵尸网络 Zeus、Agobot 和 Sdbot，分别捕获到了 5 条不同的执行轨迹。实验设计分为两部分：首先，使用 BitBlaze 的 Vine 对僵尸程序的执行轨迹进行静态分析，通过对路径约束关系的

收集、形式化表示和约束求解来分析僵尸网络的命令控制逻辑，如图 4~图 6 所示；然后，利用覆盖率分析的方法，定位僵尸网络命令控制逻辑所在的基本块，进而对基本块进行逆向分析发掘僵尸网络的控制命令，分析结果如图 7~图 9 所示。

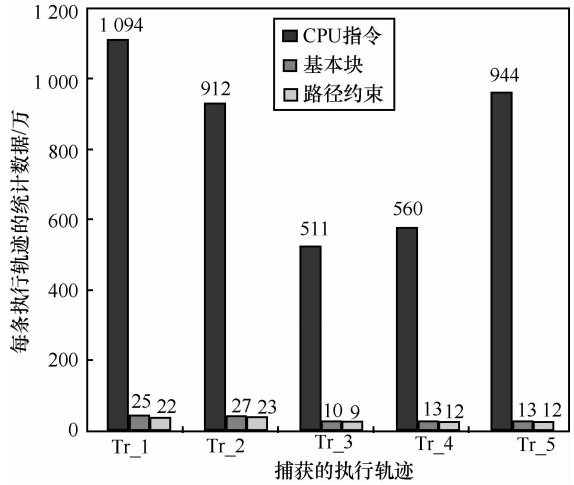


图 4　Zeus 的执行轨迹信息

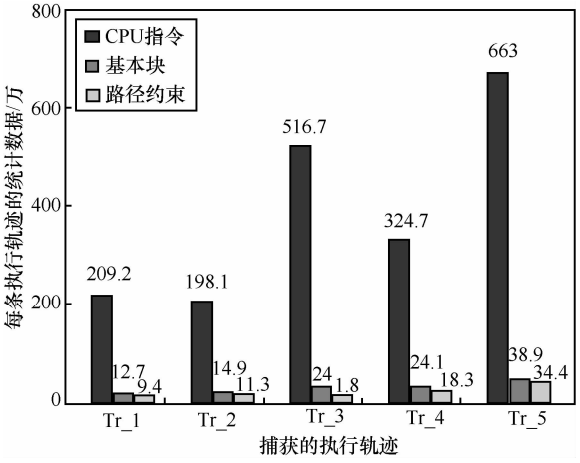


图 5　Agobot 的执行轨迹信息

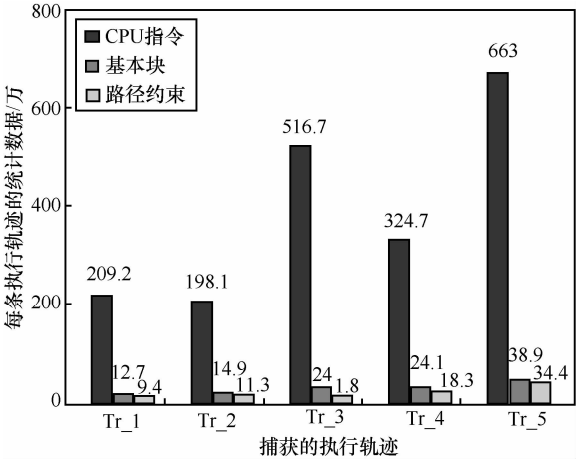


图 6　Sdbot 的执行轨迹信息

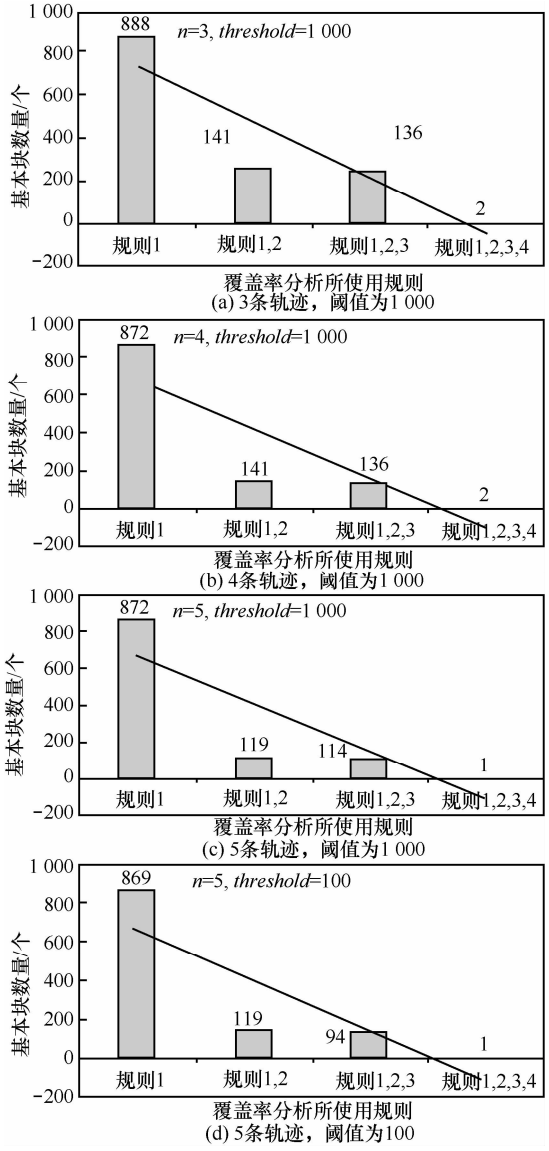


图 7　利用覆盖率分析定位 Zeus 控制命令基本块

图 4~图 6 中，包括 CPU 指令数量、基本块的数量和路径约束关系的数量。捕获的 Zeus 的 5 条执行轨迹的 CPU 指令数平均为 804.2 万，基本块的数量平均为 17.6 万，使用污点分析和符号执行去收集执行轨迹中的路径约束关系的时间开销和空间开销非常庞大。Zeus 的执行轨迹平均每条含有 15.6 万条路径约束条件，使用 Vine 对这些路径约束进行求解，能够得到有效解的只有几十个，而且这些路径约束条件与僵尸网络的控制逻辑没有直接联系。对僵尸网络 Agobot 和 Sdbot 执行轨迹的分析结果与 Zeus 的类似，分析过程需要消耗吉比特级别的存储空间和十几个小时进行约束求解，结果显示大部分路径分支的触发条件无法通过约束求解找到，有解的路径分支与僵尸网络的命令控制逻辑没有直接的联系。

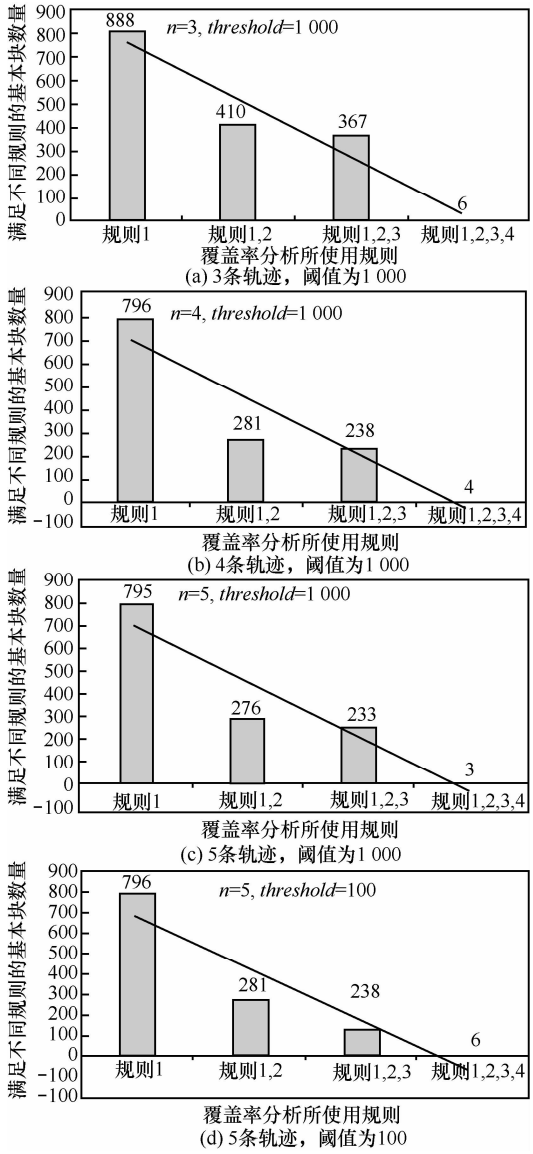


图 8　利用覆盖率分析定位 Agobot 控制命令基本块

图 7 和图 8 是利用算法 1 对 Zeus 和 Agobot 执行轨迹的分析结果。*n* 表示调用识别算法时所输入的执行轨迹的数量，实验中分别使用了 3 组、4 组和 5 组执行轨迹进行基本块的覆盖率分析，实验结果表明输入的执行轨迹数量越多，命令控制逻辑基本块的定位越准确。但是，随着执行轨迹的数量增多，必然将增加捕获执行轨迹的时间开销和空间开销。

Zeus 和 Agobot 的实验结果表明，5 条执行轨迹就可以准确定位到命令控制基本块，其需要的执行轨迹数量远小于进行路径约束条件分析时所需的执行轨迹数量。

Threshold 表示调用基于循环结构的识别算法时所输入的覆盖次数的阈值，在实验中分别将 threshold 设置为 1 000 和 100，实验结果表明阈值定位的越小，基本块的定位越准确。但是覆盖次数阈值不能小于僵尸网络控制命令的数量，如果小于则会导致没有基本块能满足所有的规则。

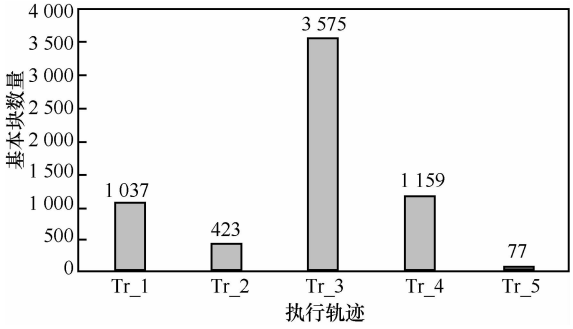


图 9　利用分支结构的覆盖率分析定位到的 if 基本块特征

僵尸网络 Sdbot 所使用的命令控制逻辑是基于分支结构的，利用算法 2 分支结构命令控制逻辑基本块识别算法对捕获的 Sdbot 执行轨迹进行覆盖率分析。通过分支结构中 if 基本块的 5 个特征，成功定位到 Sdbot 控制逻辑的位置。图 9 是对 if 基本块后面所跟的 else 基本块数量，发现不同轨迹中 if 基本块后面所跟 else 基本块的数量是不相同的，数量越多说明控制命令在控制逻辑处理过程中的位置越靠后。执行轨迹 1、3、4 中 else 基本块数量都超过了 1 000，逆向分析这些基本块发现在基于分支结构的控制逻辑的识别过程中有大量的计算代码，这些计算过程也是 if-else 结构的一部分，而且覆盖信息满足定义的规则。

利用覆盖率分析方法定位到命令控制逻辑基本块后，使用逆向分析技术可以很容易的找到僵尸网络的控制命令集合，Zeus 找到了 25 条控制命令，Agobot 中找到了 98 条控制命令，Sdbot 找到了 50 条命令。利用这些控制命令，构造了控制命令数据分组，这些控制命令所触发的执行轨迹覆盖了僵尸程序代码空间 95%以上的代码，因此可以说明覆盖率分析方法找到的控制命令集合能够全面触发僵尸程序的各种功能模块。

5　讨论

本文所提出的基于覆盖率分析的僵尸网络控制命令发掘方法是一种基于统计特征的分析方法，扩展了当前利用路径约束关系求解的推理方法，将僵尸网络命令控制协议的分析方法从路径空间的遍历扩展到代码空间的覆盖特征分析，是一种全新

的分析思路。

该基于统计特征的分析方法不需要对二进制代码进行中间表示与约束求解，分析过程的时间开销和空间开销小，并能根据代码空间是否被全覆盖来验证发现的僵尸网络命令空间是否能全面触发僵尸网络的各个功能模块，具有一定的实用性。

但是，该分析方法需要预先捕获足够数量的僵尸网络执行轨迹才可以进行统计特征分析，因此不能用于检测未知的僵尸网络。

该方法是一种对僵尸程序代码空间统计特征的分析方法，其前提条件是代码空间是可见的，而且各个代码基本块的功能性是确定的。因此，该方法会受到代码空间混淆技术的干扰，例如代码移动(code mobility)混淆技术[39]可以将部分代码变成在可控环境下是不可见的；面向返回的编程技术ROP(return-oriented programming)[38]可以动态地将代码块任意组装成不同的功能模块，破坏了代码块由其功能性所产生的统计特征。而且，僵尸网络的演化和变异速度很快，需要进一步挖掘僵尸程序执行轨迹集合对代码空间的覆盖信息中内在的本质的规律性，实现更高效地逆向构建僵尸网络的命令控制协议。

6 结束语

本文提出了一种基于覆盖率分析的僵尸网络控制命令发掘方法，对僵尸网络命令控制协议的分析提供了一种全新的思路，引入了轻量级的基于统计特征分析与识别的发掘方法。僵尸网络的执行轨迹对命令控制逻辑所在代码块的覆盖率具有一定的规律性，从中可以快速准确地定位到命令控制逻辑所在的基本块，并提取出僵尸程序可执行的控制命令集合。该方法不需要对二进制代码的路径约束关系进行形式化表示和求解，时间和空间开销小，具有实用性。利用该方法对僵尸网络 Zeus、Agobot 和 Sdbot 的执行轨迹进行分析，结果表明该方法能够快速准确地发掘出僵尸网络的控制命令集合，并验证了该命令集合所对应的执行轨迹可以覆盖僵尸程序 95%以上的代码空间。

参考文献：

[1] 诸葛建伟, 韩心慧, 周林等. 僵尸网络研究[J]. 软件学报, 2008, 19(3): 702-715.
ZHUGE J W, HAN X H, ZHOU L, *et al*. Research and development of Botnets[J]. Journal of Software, 2008, 19(3): 702-715.

[2] 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.
FANG B X, CUI X, WANG W. Survey of Botnets[J]. Journal of Computer Research and Development, 2011, 48(8): 1315-1331.

[3] 王天佐, 王怀民, 刘波等. 僵尸网络中的关键问题[J]. 计算机学报, 2012, 35(6): 1192-1208.
WANG T Z, WANG H M, LIU B, *et al*. Some critical problems of Botnets[J]. Chinese Journal of Computers, 2012, 35(6): 1192-1208.

[4] 江健, 诸葛建伟, 段海新等. 僵尸网络机理与防御技术[J]. 2012, 23(1): 82-96.
JIANG J, ZHUGE J W, DUAN H X, *et al*. Research on Botnet mechanisms and defenses[J]. Journal of Software, 2012, 23(1):82-96.

[5] NAZARIO J. DDoS attack evolution[J]. Network Security, 2008, 7: 7-10.

[6] HUSNA H, PHITHAKKITNUKOON S, PALLA S, *et al*. Behavior analysis of spam Botnets[A]. IEEE COMSWARE[C]. Bangalore, India, 2008. 246-253.

[7] FREILING F, HOLZ T, WICHERSKI G. Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks[A]. Proc of the ESORICS'05[C]. Milan, Italy, 2005.319-335.

[8] BAECHER P, KOETTER M, HOLZ T, *et al*. The nepenthes platform: an efficient approach to collect malware[A]. Proc of the RAID'06[C]. Hamburg, Germany, 2006.165-184.

[9] 金鑫, 李润恒, 甘亮等. 基于通信特征曲线动态时间弯曲距离的 IRC 僵尸网络同源判别方法[J]. 计算机研究与发展, 2012, 49(3): 481-490.
JIN X, LI R H, GAN L, *et al*. IRC Botnets' homology identifying method based on dynamic time warping distance of communication feature curves[J]. Journal of Computer Research and Development, 2012, 49(3): 481-490.

[10] GU G, PORRAS P, YEGNESWARAN V, *et al*. BotHunter: detecting malware infection through ids-driven dialog correlation[A]. Proc of the 16th USENIX Security Symp[C]. Boston, Massachusetts, USA, 2007. 167-182.

[11] GU G, PERDISCT R, ZHANG J, *et a1*. BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection[A]. Proc of the 17th USENIX Security Symp[C]. San Jose, California, USA, 2008.269-286.

[12] GU G, ZHANG J, LEE W. BotSniffer: detecting botnet command and control channels in network traffic[A]. Proc of the NDSS[C]. San Diego, USA, 2008.

[13] 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, 32(10): 1980-1988.
WANG W, FANG B X, CUI X. IRC Botnet detection based on host behavior[J]. Chinese Journal of Computers, 2009, 32(10): 1980-1988.

[14] HOLZ T, GORECKI C, RIECK C, *et al*. Detection and mitigation of fast-flux service networks[A]. Proc of the NDSS[C]. San Diego, USA, 2008.

[15] CHING-HSIANG H, HUANG C Y, CHEN K T. Fast-flux bot detection in real time[A]. Proc of the RAID[C]. Menlo Park, California, USA, 2011. 464-483.

[16] 王海龙, 龚正虎, 侯捷. 僵尸网络监测技术研究进展[J]. 计算机研究与发展, 2010, 47(12):2037-2048.
WANG H L, GONG Z H, HOU J. Overview of Botnet detection[J]. Journal of Computer Research and Development, 2010, 47(12): 2037-2048.

[17] FREILING F, HOLZ T, WICHERSKI G. Botnet tracking: exploring a root-cause methodology to prevent denial of service attacks[A]. Proc of the ESORICS[C]. Milan, Italy, 2005. 319-335.

[18] RAJAB M, ZARFOSS J, MONROSE F, *et al*. A multifaceted approach to understanding the Botnet phenomenon[A]. Proc of the 6th ACM SIGCOMM Conf on Internet Measurement[C]. Pisa, Italy, 2006. 41-52.

[19] JUAN C, PONGSIN P, CHRISTIAN K, *et al*. Dispatcher: enabling active botnet infiltration using automatic protocol reverse- engineering[A]. Proc of the CCS 2009[C]. Chicago, IL, USA, 2009. 621-634.

[20] KANICH C, KREIBICH C, LEVCHENKO K, *et al*. Spamalytics: an empirical analysis of spam marketing conversion[A]. Proc of the CCS 2008[C]. Alexandria,VA, USA , 2008. 3-14.

[21] 应凌云, 杨轶, 冯登国等. 恶意软件网络协议的语法和行为语义分析方法[J]. 软件学报, 2011, 22(7): 1676-1689.
YING L Y, YANG Y, FENG D G, *et al*. Syntax and behavior semantics analysis of network protocol of malware[J]. Journal of Software, 2011, 22(7): 1667-1689.

[22] 刘豫, 王明华, 苏璞睿等. 基于动态污点分析的恶意代码通信协议逆向分析方法[J]. 电子学报, 2012, 40(4): 661-668.
LIU Y, WANG M H, SU P R, *et al*. Communication protocol reverse engineering of malware using dynamic taint analysis[J]. Acta Electronica Sinica, 2012, 40(4): 661-668.

[23] CHO C, BABIC D, SHIN E, *et al*. Inference and analysis of formal models of botnet command and control protocols[A]. Proc of the CCS 2010[C]. Chicago, IL, USA, 2010. 426-439.

[24] KANG B, ERIC C, LEE C, *et al*. Towards complete node enumeration in a peer-to-peer Botnet[A]. Proc of the CCS 2009[C]. Chicago, IL, USA, 2009. 23-34.

[25] STONE-GROSS B, COVA M, CAVALLARO L, *et al*. Your Botnet is my botnet: analysis of a Botnet takeover[A]. Proc of the CCS 2009[C]. Chicago, IL, USA, 2009. 635-647.

[26] DAGON D, ZOU C, LEE W. Modeling botnet propagation using time zones[A]. Proc of the NDSS[C]. San Diego, USA, 2006. 235-249.

[27] VOGT R, AYCOCK J, JACOBSON M. Army of botnets[A]. Proc of the NDSS[C]. San Diego, USA, 2007. 111-123.

[28] 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.

[29] WANG W, FANG B, CUI X, *et al*. A user ID-centralized recoverable Botnet: structure research and defense[J]. International Journal of Innovative Computing, Information and Control, 2010, 6(4): 4307-4317.

[30] ZENG Y, SHIN K, HU X. Design of SMS commanded-and-controlled and P2P-structured mobile botnets[A]. Proc of the 5th ACM Conf on Security and Privacy in Wireless and Mobile Networks[C]. Tucson, Arizona, USA, 2012. 137-148.

[31] SINGH K, SENGAL S, JAIN N, *et al*. Evaluating Bluetooth as a medium for Botnet command and control[A]. Proc of the Int Conf on Detection of Intrusions and Malware, and Vulnerability Assessment[C]. Bonn, Germany, 2010. 61-80.

[32] CUI X, FANG B, YIN L, *et al*. Andbot: towards advanced mobile Botnets[A]. Proc of the 4th USENIX Workshop on Large-scale Exploits and Emergent Threats[C]. Berkeley, California, USA, 2011.11.

[33] ZHAO S, LEE P, LUI J, *et al*. Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service[A]. Proc of the Annual Computer Security Applications Conf (ACSAC 2012)[C]. Florida, USA, 2012.119-128.

[34] AMINI P, PIERCE C. Kraken Botnet infiltration[EB/OL]. http://dvlabs. tippingpoint.com.

[35] CHIA Y, JUAN C. Botnet infiltration: finding bugs in botnet command and control[EB/OL]. http://www.eecs.berkeley.edu/~chiayuan/cs261/cs261_cho.pdf, 2009.

[36] DAVIS C, FERNANDEZ J, NEVILLE S, *et al*. Sybil attacks as a mitigation strategy against the storm Botnet[A]. Proc of the 3rd Int Conf on Malicious and Unwanted Software[C]. Alexandria, Virginia, USA, 2008. 32-40.

[37] BARFORD P, YEGNESWARAN V. An inside look at Botnets[J]. Malware Detection, 2007,27: 171-191.

[38] SHACHAM H. The geometry of innocent flesh on the bone: return-into-libc with-out function calls (on the x86)[A]. Proc of the 14th ACM Conf on Computer and Communications Security[C]. Alexandria, VA, USA, 2007.552-561.

[39] FALCARIN P, CARLO S, CABUTTO A, *et al*. Exploiting code mobility for dynamic binary obfuscation[A]. Proc of the World Congress on Internet Security[C]. London, UK, 2011. 114-120.

[40] CUI W, KANNAN J, WANG H J. Discoverer: automatic protocol reverse engineering from network traces[A]. Proc of 16th USENIX Security Symposium on USENIX Security Symposium[C]. Boston, MA, USA, 2007. 1-14.

[41] VIGNA G. Static disassembly and code analysis[J]. Malware Detection, 2007,27:19-41.

[42] CABALLERO J, POOSANKAM P, KREIBICH C, *et al*. Bidirectional Protocol Reverse Engineering: Message Format Extraction and Field Semantics Inference[R]. 2009.

[43] LIM J, REPS T. BCE: Extracting Botnet Commands from Bot Executables[R]. 2010.

[44] 王志，贾春福，鲁凯. 基于环境敏感分析的恶意代码脱壳方法[J]. 计算机学报, 2012, 35(4): 693-702.
WANG Z, JIA C F, LU K. Malicious hidden-code extracting based on environment-sensitive analysis[J]. Chinese Journal of Computers, 2012, 35(4): 693-702.

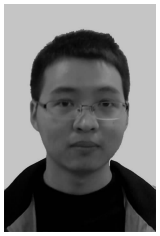
[45] Qemu[EB/OL]. http://wiki.qemu.org, 2013.

[46] SONG D, BRUMLEY D, YIN H, *et al*. BitBlaze: a new approach to computer security via binary analysis[A]. Intl Conf on Information Systems Security(ICISS 2008)[C]. Hyderabad, India, 2008. 1-25.

[47] Pin: a dynamic binary instrumentation tool[EB/OL]. http://software. intel. com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool, 2013.

作者简介：

王志（1981-），男，山西长治人，博士，南开大学讲师，主要研究方向为恶意代码的分析与防治、二进制代码混淆。



蔡亚运（1989-），男，河南周口人，南开大学硕士生，主要研究方向为恶意代码的分析与防治。

刘露（1988-），男，河南焦作人，南开大学硕士生，主要研究方向为恶意代码的分析与防治。

贾春福（1967-），男，河北文安人，博士，南开大学教授、博士生导师，主要研究方向为信息安全与可信计算、恶意代码发现与分析。

（上接第 155 页）

[10] XIA Y, SUBRAMANIAN L, STOICA I, *et al*. One more bit is enough[J]. Proceedings of ACM Transactions, 2005,16(6):1281-1294.

[11] CHALLET D, ZHANG Y C. Emergence of cooperation and organization in an evolutionary game[J]. Physica A: Statistical Mechanics and its Applications, 1997,246(3-4):148-407

[12] KUTSUNA H, TAGASHIRA S, FUJITA S. A fair and efficient congestion control scheme based on minority game[A]. Proc of the 14th IEEE International Conference on Networks[C]. 2006.98-103.

[13] KALINOWSKI T, SCHULTZ H J, BRISES M, Cooperation in the minority game with local information[J]. Physica A: Statistical Mechanics and its Applications, 2000,277(3-4):502-508.

[14] MISRA V, GONG W B, TOWSLEY D. A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED[J]. Computer Communication Review, 2000,30:151-160.

作者简介：

王祖喜（1964-），男，湖北武汉人，博士，华中科技大学副教授、硕士生导师，主要研究方向为网络与信息安全、模式识别与智能系统、计算智能与复杂系统理论。

邓昭彰（1988-），男，湖南邵阳人，华中科技大学硕士生，主要研究方向为计算机网络信息安全。

李力（1986-），男，湖北黄冈人，华中科技大学硕士生，主要研究方向为计算机网络信息安全。