doi :10.3969/j.issn.1671-1122.2020.11.005

基于 Rootkit 隐藏行为特征的 Linux 恶意代码取证方法

文伟平，陈夏润，杨法偿
（北京大学软件与微电子学院，北京 102600）

摘　要：近年来，在互联网不断发展的同时，网络安全问题也层出不穷，而在对抗网络安全威胁时，取证问题一直是个难题。尤其是针对 Linux 平台，目前主流的 Linux 开源取证工具多数存在滞后、效率低、无法对隐蔽性强的木马进行取证等问题。在 Linux 取证研究中，Rootkit 木马具有隐蔽性强、危害性大的特点，传统检测方法难以进行有效检测。为解决上述问题，文章从 Rootkit 的行为和实现技术出发，对其启动机制和内存驻留机制进行研究分析，提炼恶意代码行为作为检测特征，提出一种基于 Rootkit 隐藏行为特征的 Linux 恶意代码取证方法。实验表明，文章提出的取证方法对各类 Linux 恶意代码具有很好的检出效果和取证效果，相较传统取证方法在检测效果上具有明显优势。

关键词：计算机取证；Rootkit；恶意代码；Linux 系统

中图分类号：TP309　**文献标志码：**A　**文章编号：**1671-1122（2020）11-0032-11

中文引用格式：文伟平，陈夏润，杨法偿 . 基于 Rootkit 隐藏行为特征的 Linux 恶意代码取证方法 [J]. 信息网络安全，2020，20（11）：32-42.

英文引用格式：WEN Weiping, CHEN Xiarun, YANG Fachang. Malicious Code Forensics Method Based on Hidden Behavior Characteristics of Rootkit on Linux[J]. Netinfo Security, 2020, 20(11): 32-42.

Malicious Code Forensics Method Based on Hidden Behavior Characteristics of Rootkit on Linux

WEN Weiping, CHEN Xiarun, YANG Fachang
(*School of Software and Microelectronics*, *Peking University*, *Beijing* 102600, *China*)

Abstract: In recent years, with the continuous development of the Internet, network security problems emerge endlessly. When fighting against network security threats, forensics has always been a big problem. Especially for Linux platform, most mainstream Linux open source forensics tools are currently lagging behind, inefficient and unable to obtain evidence from the hidden Trojans. In the research of Linux forensics, because the Rootkit Trojan has the characteristics of strong concealment and great harm, traditional detection methods are difficult to carry out effective detection. In order to solve the above problems, starting from the behavior and implementation technology of Rootkit, this paper studies and analyzes its

收稿日期：2020-7-8
基金项目：国家自然科学基金 [61872011]
作者简介：文伟平（1976—），男，湖南，教授，博士，主要研究方向为系统与网络安全、大数据与云安全、智能计算安全；陈夏润（1997—），男，江西，硕士研究生，主要研究方向为网络与系统安全、漏洞挖掘；杨法偿（1995—），男，河南，硕士研究生，主要研究方向为系统安全、计算机取证。
通信作者：文伟平 weipingwen@ss.pku.edu.cn

startup mechanism and memory resident mechanism, extracts malicious code behaviors as detection features, and proposes a Linux malicious code forensics method based on Rootkit hidden behavior characteristics. The experimental results show that the forensics method proposed in this paper has a good detection effect and forensics effect for various types of Linux malicious code, and has obvious advantages in detection effect compared with traditional forensics methods.

Key words: computer forensics; Rootkit; malicious code; Linux system

0 引言

计算机取证技术是伴随着网络攻击频繁发生而衍生的一门科学，是对恶意攻击的事后检测技术。当系统被攻击者攻陷后，快速有效地从受害主机中发现攻击者木马以及留下的相关痕迹至关重要。由于计算机证据的特殊性，如脆弱性、不可靠性等，导致对计算机证据的合法性和可靠性一直存在质疑。根据DAFOULAS[1]等人对英国、美国计算机取证项目的回顾和调研可以发现，近年来随着越来越多的国家对计算机证据合法性的确认，计算机取证技术迎来了快速发展。

目前针对Linux平台的取证多为取证人员手动查找可疑信息并结合木马检测工具进行取证，取证效率低、取证速度慢，Rootkit技术的发展更是增加了Linux系统的取证难度。斯添浩[2]等人通过对Linux平台常见的Rootkit隐藏与检测技术进行研究，提出Rootkit木马本身具有隐蔽性强、难发现的特点，经常被攻击者用来隐藏攻击痕迹和建立后门，危害性极大，特别是内核层Rootkit木马侵入内核后拥有了root级权限，使得恶意代码破坏性更强，且较难检测。随着Rootkit技术的发展，其躲避检测的技术愈发多样，隐藏形式也愈发偏向内核层，传统的木马取证手段的检测能力逐渐变弱，Rootkit取证已成为众多取证与木马检测工具的痛点和难点[3]。

为了解决上述问题，本文从Rootkit恶意代码的行为和实现技术出发，研究Rootkit启动机制和内存驻留机制，提炼其隐藏行为作为检测特征，提出一种基于Rootkit隐藏行为特征的Linux恶意代码取证方法，对恶意代码进行取证与检测，快速定位恶意信息，提高取证效率。

1 相关工作

目前国内外计算机取证技术迅速发展，在取证模型、取证方法上不断涌现出新的思路，如对于Linux系统的取证方法研究[4-6]，对于勒索软件的取证分析[7]，对于图片隐写、数据隐藏技术的取证研究[8,9]，以及对于各种电子证据的取证[10,11]等。Rootkit作为一种特殊的恶意软件，可以通过修改系统行为对自己的关键信息进行隐藏[12]，取证难度大，且Linux平台的Rootkit取证更是一大难题。目前国内外对Rootkit的取证主要分为软件取证法与硬件取证法两大类别。

1.1 Rootkit 软件取证法

Rootkit软件取证法比较常见，主要是对系统的特征、状态、视图等进行分析校验，以发现隐藏在系统中的Rootkit恶意代码。例如，梁升荣[13]等人提出一种通过分析内核模块加载时的可疑行为，结合System.map和kmem文件进行对比的Rootkit检测方法。该方法可以描述一定的恶意代码行为特征，但是该方法仅对内核加载模块进行分析，无法对复杂的恶意代码进行取证。颜仁仲[14]等人提出一种利用System.map文件进行Rootkit检测和受损系统恢复的方法，该方法存在和文献[13]类似的问题。MUSAVI[15]等人提出一种静态分析检测Rootkit的方法，通过对内核驱动程序中的相关特征进行提取对Rootkit行为进行刻画，从而检测是否存在恶意代码。陈远鹏[16]等人提出一种内核层Rootkit检测方法，通过交叉对比从不同层面获取的虚拟机用户态视图、内核态视图和VMM层视图来进行取证，效果较好。但是，这种检测技术太过依赖宿主机操作系统和虚拟机监视器，难以进行实际应用。在系统完整性校验上，FANG[17]等

人提出一个分布式环境下对虚拟机的完整性进行监控和检测的系统VMGuard，主要在每个物理节点上运行一个特殊的VM以监视同时运行的VM管理机，将形成的虚拟机保护域收集的完整性数据发送到VMGuard服务器进行安全存储和独立分析，以进行Rootkit检测。BEHROZINIA[18]等人提出一种通过使用虚拟机监视器（VMM）监视内核数据访问来阻止和检测数据内核Rootkit攻击的方法，该方法在内核Rootkit的检测上具有较好的效果，但是检测方式过于单一，无法应对一些内核数据活动较复杂的Rootkit恶意代码。

机器学习与深度学习也逐渐应用到基于软件特征的检测上。LUCKETT[19]等人提出将神经网络用于分析未感染和感染Rootkit的系统行为，对系统行为通过二分类的方式进行检测。LOBO[20]等人对各种Rootkit进行动态分析，使用sLOPE聚类算法将这些Rootkit聚类到多个类别，并使用迭代二分法生成决策树，用于识别计算机是否感染Rootkit。JOY[21]等人提出一种基于*k*-means聚类算法的内核级Rootkit检测方法。通过机器学习进行Rootkit取证效率较高，但是这类利用机器学习进行检测的方法更多依赖于现有恶意代码集，对于未知恶意代码和隐藏木马的取证效果不佳。

随着Rootkit检测技术的不断发展，对于Rootkit实现原理的研究也在不断加深。例如，冯培钧[22]等人分析了当前主流Rootkit检测方法能够检测出传统内核级Rootkit的原因，并在对Linux内核工作机制和数据结构进行深入研究和分析的基础上，设计并实现了一种Linux新型内核级Rootkit，该Rootkit能够绕过主流的Rootkit检测方法，这也启发了研究者在进行Rootkit检测研究时需要深入分析Rootkit的实现原理及隐藏机制。

1.2 Rootkit 硬件取证法

除了Rootkit软件取证法外，还有很多利用硬件特性[23,24]进行Rootkit取证的方法。Rootkit硬件检测工具CoPilot[25]能够通过对系统进行硬件级别的监控查找系统中的异常行为。除此之外，AKAO[26]等人提出一种检测内核Rootkit的方法KRGuard，该方法可以监视内核空间中的分支记录。许多内核Rootkit生成的分支与内核空间中常见的分支不同，KRGuard可以通过使用普通处理器的硬件特性来检测这些差异。MOON[27]等人提出一种通过总线嗅探来防止内核Rootkit攻击的方法。除了上述方法外，还有Tribble[28]及Fred[29]两种硬件检测方法，它们通过对物理内存的复制来检测是否存在异常行为。

从安全角度看，硬件检测法的安全性高于软件检测法，但是由于实现的复杂性以及对硬件的依赖性，Rootkit硬件检测法的研究依然任重道远。

综上，国内外现有Linux Rootkit恶意代码取证方法多数存在效率低、实现复杂、条件依赖性高的特点。根据对现有相关研究的分析可知，需要从Rootkit的隐藏原理出发，结合Linux系统各层次关键信息特征对恶意代码进行取证。

2 Rootkit 行为特征关键数据提取

在取证中，对于关键信息的提取和备份极为重要。针对Rootkit隐蔽性强、难以被检测的问题，本文将对隐藏方式的分析转变为对恶意代码隐藏行为的分析，从而大大提高检测效率及准确度。在对隐藏行为的分析中，首先需要对系统用户层、内核层、流量层的关键信息进行提取，再使用本文提出的取证方法进行数据分析，进而得出反映恶意代码的行为特征，实现恶意代码取证。

通过对Rootkit启动机制和内存驻留机制进行分析，发现涉及的关键数据主要包含以下几个部分：

1）内存与元数据文件。目标主机的内存必须为提取的第一个关键数据，因为主机内存会因为后续操作而改变，影响内存状态。

2）磁盘信息。将目标主机的整个磁盘信息提取出来。

3）系统信息。包括系统版本、内核版本、登录用户、运行时间、内存状态、磁盘状态等，此类信息可以帮助进行系统环境的复现。

4）用户文件信息。系统登录信息/etc/shadow、/etc/

passwd中的文件信息，以及相应的原数据信息、组信息等。

5）进程信息。系统中正在运行的所有进程的状态信息、进程打开的文件信息。

6）网络信息。系统中正在运行的所有网络连接信息，包括开放端口、监听端口、DNS缓存、ARP缓存、流量信息等。

7）定时任务信息。系统中正在运行的定时任务信息，包括crontab命令执行信息、etc/cron.* 文件信息。

8）启动项信息。系统中正在运行的自启动服务文件信息。

9）内核模块信息。系统中已加载的内核模块、相应的文件信息以及加载日志、模块详情信息等。

10）服务信息。系统中可运行的服务信息，包括chkconfig命令执行信息和rpcinfo命令执行信息。

11）完整性信息。常用系统命令（如ls、netstat、ps等）执行结果的完整性信息。

12）关键文件信息。环境变量信息、/tmp文件夹下文件、特权suid文件、/bin文件夹下文件、/sbin文件夹下文件、.rhosts和.forward等.*形式隐藏文件。

13）日志文件信息。系统登录日志、命令行日志、syslog日志、启动日志、dmesg日志、防火墙日志、浏览器日志、/var/log日志等日志文件信息。

3 基于 Rootkit 隐藏行为特征的取证方法

无论黑客使用何种方式对系统进行修改，目前的Rootkit恶意代码的最终目的都是隐藏文件、网络连接、内核模块、进程等关键信息。基于这个特性，本文忽略恶意代码具体隐藏方式，而重点关注隐藏行为。对Rootkit隐藏行为特征相关的关键信息进行提取后，对数据进行分析，提取Rootkit隐藏行为特征，从而设计出基于Rootkit隐藏行为特征的取证方法。本文设计的取证方法将关键信息主要划分为用户层、内核层和流量3个方面进行分析，取证方法总体框架如图1所示。

本文设计的取证方法提取出的Rootkit隐藏行为特征包括隐藏文件、隐藏网络连接、隐藏进程、隐藏模块、隐藏端口和异常流量等，这些特征通过分析用户层关键数据、内核层关键数据及流量而得出。下面进行详细描述。

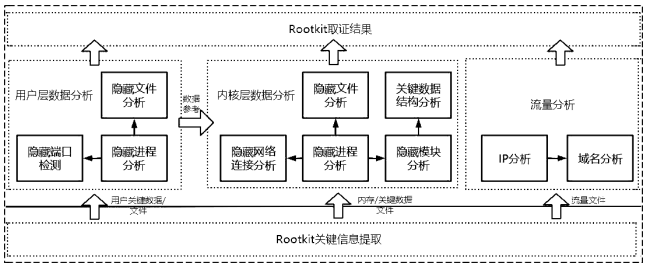


图 1 取证方法总体框架

3.1 用户层关键数据分析

用户层的关键数据分为3类：第一类是确定的恶意软件所产生的恶意数据，包括恶意进程、恶意文件、恶意网络连接等；第二类为确定的白样本所产生的正常数据；除此之外的都为第三类可疑数据。针对这些关键数据，本文提出如图2所示的分析流程。

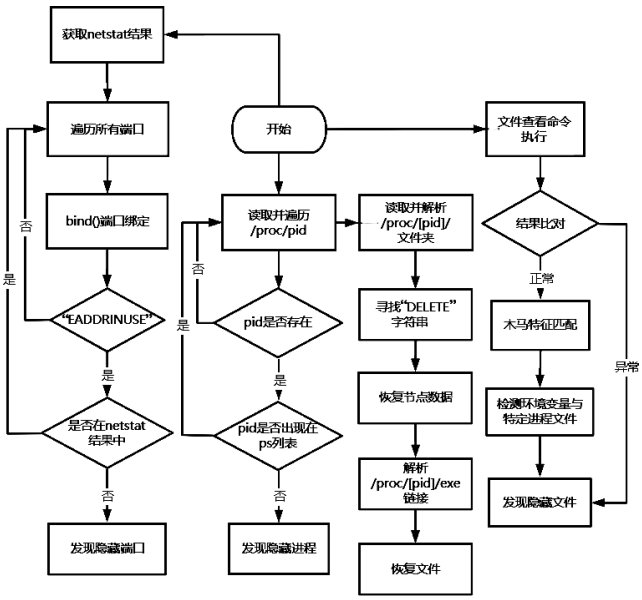


图 2 用户层关键数据分析流程

具体描述如下：

1）利用Linux服务器端口在不开启端口复用情况下只能被唯一进程进行绑定的特性进行隐藏端口检测。利用C语言中的函数bind()对端口进行绑定，对所有端口进行遍历，根据函数返回状态判断端口是否开启。若端口绑定成功，则证明端口未被其他进程占用。若端口绑

定失败，bind()返回“EADDRINUSE”，证明此端口已被其他进程占用，继续判断端口号是否在网络状态查看命令netstat的查询结果中。若在,证明端口未被隐藏；反之，证明端口被隐藏。

2）根据Linux进程在系统中的运行特点，通过检测进程属性来验证进程是否存在，并和利用系统自带命令行工具对关键数据进行提取得到的基本取证进程信息进行对比，发现隐藏进程。读取并遍历/proc/pid中存储的进程标识号，获取系统中存在的所有进程的pid进程标识号。对每个存在的进程信息进行分析，与当前进程列举命令ps的执行结果进行对比，判断是否属于隐藏进程。此外，读取并解析存储在proc文件系统中的文件信息，根据字符串“DELETE”寻找被木马删除的文件数据，对找到的文件数据链接进行解析，恢复被木马删除的文件。

3）对比基本取证与深度取证中的文件查看命令（ls、find等）的执行结果。如果存在不一致，则存在差异的文件即为Rootkit隐藏文件。如果比对结果正常，则与已知木马特征进行对比，判断是否存在已知木马运行过程中存在的特定隐藏文件，从而完成对已知木马的定位检查，发现已知木马的Rootkit隐藏文件。

3.2 内核层关键数据分析

随着Rootkit技术的不断发展与进步，攻击者越来越倾向于实现内核层Rootkit的隐藏，原因有二：一是内核层Rootkit隐藏不易被发现；二是攻击者在内核层面可以获得更高的权限，可以实施更多的恶意功能。因此对内核层的检测极其重要。

同用户层检测一样，在内核层面同样不关注攻击者的具体隐藏方式，而是关注攻击者的隐藏内容。不管以何种方式进行隐藏，攻击者的最终目的均是隐藏文件、隐藏进程、隐藏网络信息、隐藏内核模块，而所有隐藏的信息，要想在系统中存活，必然以某种方式在内存中存在。因此本文基于内存枚举信息与基本取证信息交叉对比的方式对内核层的关键数据进行分析，从而对Linux系统恶意代码隐藏行为进行检测和取证。

本文提出的内核层关键数据分析流程如图3所示。

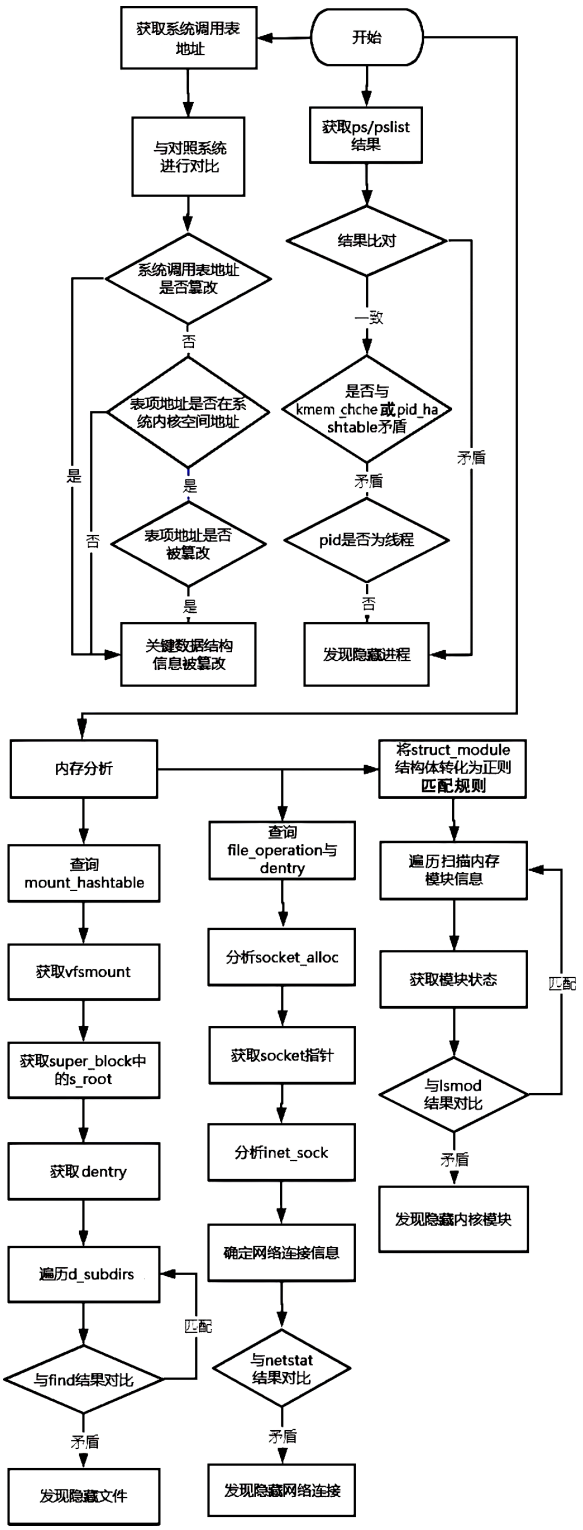


图 3 内核层关键数据分析流程

具体描述如下：

1）基于内核系统调用表分析关键数据结构信息是否被篡改。传统对系统调用表的检测多是基于System.map文件中的系统调用表，随着Rootkit技术的发展，System.map文件已经可被篡改，这种篡改使得System.map变得不可信，从而导致传统检测方法的失效。本文方法通过获取测试系统中的系统调用表地址，将系统调用表地址与无木马的对照系统中System.map产生的系统调用表地址进行比较。如果不一致，则说明系统调用表被篡改；如果一致，则继续对系统调用表的表项地址进行分析，分析表项起始地址是否在系统内核空间地址中。若不在，则说明系统调用表被篡改；若在，则将系统调用表表项地址与System.map表项地址进行比较，判断表项地址是否被篡改，若地址不一致，则说明表项地址被篡改，即系统调用表已被Rootkit木马篡改，也即内核关键数据结构信息被篡改。

2）基于多种进程视图信息对比对隐藏进程进行分析。首先将用户层ps命令结果与pslist命令结果进行比较，如果不一致，则证明为隐藏进程；如果一致，则继续判断是否与kmem_cache内存信息或pid_hashtable进程表一致。若与kmem_cache内存信息或pid_hashtable进程表都一致，则说明该进程为正常进程；若与两者中的任意一个不一致，则说明该进程为可疑进程。由于kmem_cache和pid_hashtable 中均包含线程信息，因此需要分析该可疑进程的pid标识的是否是一个线程。若是，则不是隐藏进程；若不是，则是隐藏进程。

3）基于内存文件信息枚举对隐藏文件信息进行分析。目前内核层针对文件信息的隐藏均是通过Hook掉VFS文件系统的常用函数来阻止内存中文件相关信息在Linux文件系统的显示，但并没有隐藏文件在内存中的数据结构和元数据信息。因此只要能枚举内存中的文件信息，就可以发现隐藏的文件信息。Linux系统在内核内存中维护了一个文件系统列表mount_hashtable，通过此表可以找到存储了所有挂载在系统中的文件系统的结构vfsmount；同时可以从vfsmount中获得super_block结构，进而使用super_block中的成员s_root获取文件系统的根dentry；再递归遍历denrty中的列表d_subdirs来获取所有文件和列表，并将遍历结果与find命令的结果进行对比，判断内存恢复的文件是否可在用户层中找到，以完成对隐藏文件的查找。

4）基于内存恢复隐藏网络连接信息。网络连接信息是Linux系统通过seq_file接口写入/proc中的信息。通常都是通过替换掉接口操作中的函数show()来对想要隐藏的端口与网络连接进行隐藏。由此可见，网络连接信息在内存中均是处于一个完整的状态，只要能枚举内存中的网络连接信息，就可以与基本取证模式中的网络连接信息进行对比，从而发现隐藏的网络连接信息。本文通过查询file_opration和dentry结构查询所有socket连接的socket_alloc结构，从中获取socket指针，进而对socket的派生类inet_sock进一步分析以获取socket对应的网络连接信息，如IP地址、端口等，并与netstat命令结果进行对比，根据结果是否一致判断是否存在隐藏网络连接。

5）基于内存模块结构实例发现隐藏模块信息。当系统使用函数insmod()或者modprobe()将内核模块加载进内核时，需要优先调用函数init_module()。在调用init_module()时，通过函数load_module()在内核空间对加载的内核模块文件进行解析，创建一个struct_module结构体，并在内核与内存中用此结构体代表该内核模块。只要内核模块在内存中存在，属于该模块的结构体实例一定存在。本文提出将struct_module结构体转化为正则匹配规则，并使用该匹配规则对内存进行遍历扫描，获取系统中运行的内核模块信息与状态，将结果与lsmod命令结果进行对比，根据结果是否一致判断是否存在隐藏的内核模块。

3.3 流量分析

目前工业界对流量的分析主要侧重于对网站流量进行分析，通过流量大小、活跃IP数量、网站日访问量、网站页面大小、网站页面个数等网站业务层数据对网站流量情况进行描述。但是将这些分析方法应用于木

马分析时，分析出来的结果与木马实际在系统中产生的流量并不一致，存在较大的偏差。因此本文提出一种从IP承载域名情况、IP探测端口开放情况、IP（域名）对应页面大小情况、IP（域名）对应网站连接个数情况等方面综合考虑来衡量IP可疑程度的方案，将IP可疑程度抽象为IP恶意指数来进行计算，如公式（1）所示。

$$MD=\frac{P}{1000}+\frac{L}{10}+\frac{(100-D)}{10}+(10-PT) \quad (1)$$

其中，P 为页面大小，L 为网站连接个数，D 为IP承载域名数量，PT 为端口开放数量。当 P 和 L 变大时，会增大 MD。当 D 小于100且在增大时，会增大 MD；当 D 增大到大于100时，会逐渐减少 MD。当 PT 在10以下时，对 MD 有增益；否则，会减少 MD。

流量分析后将获取一些可疑的IP及域名信息，对这些可疑的IP及域名信息进行解析，检测是否为恶意域名，并根据域名分析获取更多关于恶意木马的信息。

在域名解析方面，很多恶意软件回连的域名都是相同的，同家族恶意软件回连域名的相似度也很高。因此可以将流量包中HTTP协议访问的域名与DNS协议包中解析的域名进行提取，与恶意域名库进行对比，判断有无域名处于恶意域名库中。如果发现恶意域名，则进一步确定恶意代码类型，帮助缩小取证范围与取证目标，加快取证速度与效率。

除了将域名与恶意域名库对比之外，域名的混乱程度对恶意域名定位也很有帮助。通过对现有的一些恶意域名库进行分析发现，较多的恶意域名都是随机生成的，字符混乱程度较高。例如，门罗比挖坑木马的回连域名为jw-js1.ppxxmr.com，与正常域名如www.baidu.com相比，混乱程度要高。因此可以对域名混乱程度进行计算，具体可采用字符熵的方式，字符熵计算方法如公式（2）所示。其中，$E(C)$ 为域名 C 的混乱程度，N 为 C 所含ASCII码总数。混乱程度越高，IP可疑程度越大。

$$E(C)=-\sum_{i=1}^{128}(\frac{C_i}{N})\log(\frac{C_i}{N}) \quad (2)$$

$$N=\sum_{i=1}^{128}C_i \quad (3)$$

4 实验及分析

为测试本文方案的取征效果及针对Rootkit的适用性，本文选用的研究样本均来源于近年来比较活跃的Linux木马样本以及几种未公开保密级的木马样本。由于本文取证方法能够发现Rootkit隐藏型木马，因此实验选择的大部分木马均伴随不同层次的Rootkit隐藏。例如，Billgates木马存在用户层Rootkit隐藏，Reptile木马伴随内核层Rootkit隐藏，Ddrk木马具有混合型Rootkit隐藏等。具体实验选择木马情况如表1所示。

表 1 实验选择木马

木马名称	木马简介
Remaiten	拥有扫描器、下载器和后门功能，目的在于入侵和控制一些运行的 Linux 系统目标，部署僵尸网络
Billgates	近年来较活跃的 DDoS 僵尸网络木马，通过部署木马所控制的傀儡机壮大僵尸网络
Reptile	Github 上的 LKM Rootkit，经常被黑客使用进行 Rootkit 隐藏
WNPS	Adore-ng 木马的加强版，使用加密通信，在 Linux 2.6.x 内核中通用，隐蔽性很高
Ddrk	结合 shv 木马和 adore-ng 木马的特点、内核级别的 Rootkit 木马

除表1之外，实验还收集了两种未公开木马，以测试本文方法对未知恶意代码的取证效果。

4.1 不同类别木马取证结果对比

取证工作是事后检测机制，因此实验环境要尽量与现实情况保持一致。具体实现方式是先将木马在目标主机虚拟机中分别进行部署，每个木马环境独立。待木马启动成功后，再将取证方法中关键数据提取模块与用户层数据检查模块部署到目标主机虚拟机中，先进行关键数据提取，然后进行用户层数据检查。两个模块均运行完毕后，再将结果打包，利用网络传输工具传输到取证主机，并对运行木马的目标主机进行内核层分析与流量分析。具体检测结果如表2所示，其中Unknow1与Unknow2是实验收集的两种未公开木马。

由表2可知，本文方法对各类型木马均具有较好的检测效果，对不同层次的Rootkit隐藏木马能够使用对应的检测模块进行检测取证。

为验证本文方法的木马检出情况，选取Linux常用的木马取证工具Rkhunter-1.4.6和ChkRootkit-0.52进行对

表 2 各类木马取证结果

木马名称	用户层检测结果	内核层检测结果	流量分析	是否检出木马
Remaiten	隐藏文件	无检测结果	恶意域名	是
Billgates	隐藏进程、隐藏文件、隐藏网络连接	隐藏进程、隐藏文件、隐藏网络连接	恶意 IP	是
Reptile	隐藏端口	隐藏文件、隐藏网络连接、隐藏内核模块	无检测结果	是
WNPS	无检测结果	隐藏文件、隐藏进程、隐藏内核模块、关键数据结构篡改	无检测结果	是
Ddrk	隐藏进程	隐藏网络连接、隐藏进程、隐藏文件、关键数据结构篡改	无检测结果	是
Unknow1	隐藏端口	隐藏网络连接、隐藏文件、隐藏内核模块	恶意域名	是
Unknow2	无检测结果	隐藏内核模块、隐藏进程、关键数据结构篡改	无检测结果	是

比实验，木马检出情况如表3所示。可以发现，本文方法对于隐藏木马的检出情况好于其他两种取证工具。

表 3 木马检出情况对比

木马取证工具	Remaiten	Billgates	Reptile	WNPS	Ddrk	Unknow1	Unknow2
Rkhunter	√	√	×	√	√	×	×
ChkRootkit	√	√	√	√	×	×	×
本文方法	√	√	√	√	√	√	√

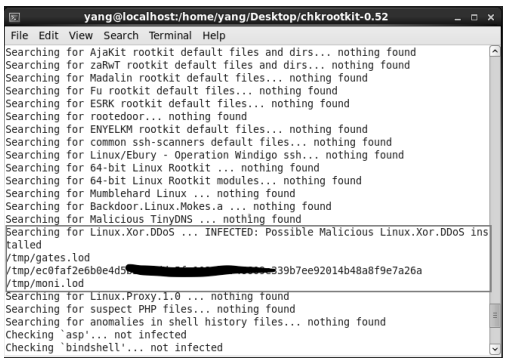
4.2 与现有取证方法效果对比

在计算机取证中，除了检出情况外，取证效果也至关重要。实验针对用户层Rootkit木马Billgates和内核层Rootkit 木马Reptile，将本文方法的取证效果与Rkhunter和ChkRootkit的取证效果进行了对比分析。

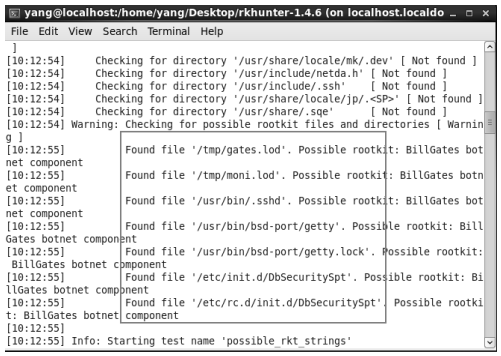
1）用户层Rootkit取证效果

Billgates木马的特点是具有恶意后门程序，能发动DDoS攻击，且会通过替换常用的系统工具进行伪装。该木马得名于其在变量函数的命名中大量使用“Gates”这个单词。公开资料显示，Billgates木马是近几年比较活跃的木马，主要针对中国地区的服务器进行DDos攻击，即95%的攻击目标是在中国。

Billgates木马部署完毕后，首先使用ChkRootkit与Rkhunter进行检测，检测结果如图4所示。由图4可知，ChkRootkit可以根据木马特征发现tmp/文件夹下木马文件以及木马执行产生的相关日志文件；Rkhunter则发现了更多信息，如可执行二进制文件、系统启动项等，但都是一些文件信息，对于进程网络连接信息等无法检出。



a）ChkRootit 检测结果



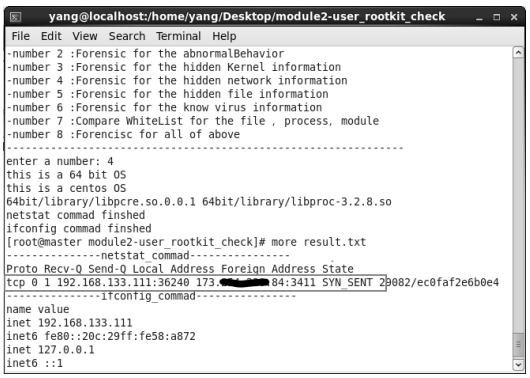
b）Rkhunter 检测结果

图 4 对比取证工具用户层 Rootkit 检测结果

对于本文方法来说，由于Billgates是用户层Rootkit木马，因此在关键数据提取完成之后，利用用户层数据检查模块即可发现隐藏信息。隐藏进程检测中，发现Billgates开启的隐藏进程/usr/bin/bsd-port/getty，并且通过lsof命令的执行结果发现该进程还开启了TCP连接。通过进程白名单比较，还另外发现了Billgates的启动信息以及Billgates的守护进程.sshd。在隐藏网络连接检测时，发现Billgates正在与服务器173.X.X.84进行通信。具体检测结果如图5所示。

采用本文方法进行流量分析的情况如图6所示。由图6可知，Billgates连接的服务器为美国加利福尼亚圣何塞的主机，承载域名最新解析为lzj.passwd1.com 和fk.appledoesnt.com。从名称看，均较为可疑，但是域名均未承载具体网站，无法进一步进行溯源分析。

a）隐藏进程相关信息检测结果



b）隐藏网络连接相关信息检测结果

图 5 本文方法用户层 Rootkit 检测结果

IP	area	inputSize	outputSize	Domains	DomainsNum	Port	PageSize	LinkNumbers	Comprehensive
173.X.X.84	美国加利福尼亚圣何塞	1457	4890	2020-02-29 00:38:54 : lzj.passwd1.com 2020-01-09 03:28:49 : fk.appledoesnt.com	7	22,443,	0	0	16

图 6 本文方法流量分析情况

至此，本文方法发现了Billgates存在于目标主机中的绝大多数关键信息，足以清晰地确认Billgates的存在，取证效果较传统工具明显提高。

2）内核层Rootkit取证效果

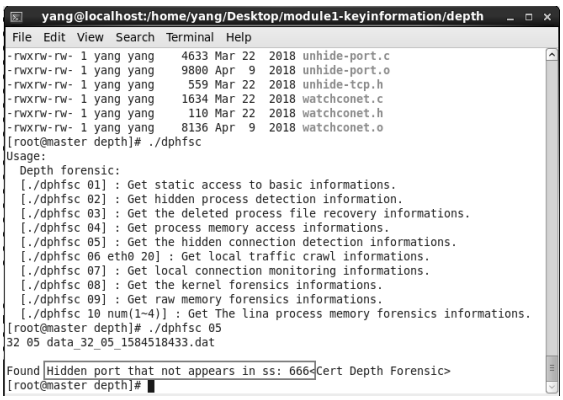
Reptile是GitHub上一个活跃的LKM类型Rootkit，功能强大，不仅支持隐藏文件、隐藏进程、隐藏端口、隐藏内核模块，还支持对用户提权和建立后门等各种恶意功能。近年来更新频繁，能够运行在目前市场上比较流行的各类Linux操作系统与内核上，是黑客经常使用与改造的内核层Rootkit木马。

Reptile安装成功后，会隐藏自身文件与加载的内核模块。利用Reptile在目标主机的666端口建立供恶意攻击者连接的后门，并对监听网络连接进行隐藏。Reptile木马部署完毕后，利用本文方法和ChkRootkit对目标主机进行检测。ChkRootkit检测结果如图7所示。

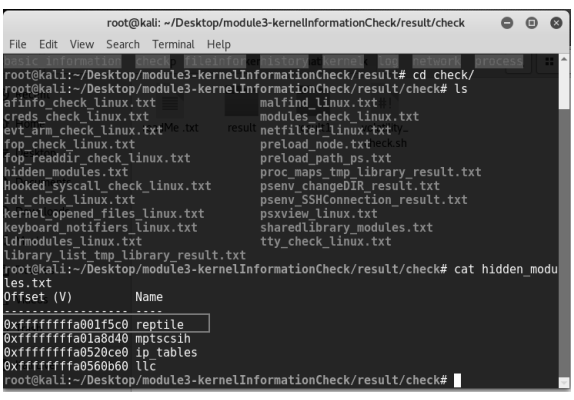
图 7 ChkRootkit 内核层 Rootkit 检测结果

从图7可以看出，ChkRootkit只检测出存在LKM类型木马，具体木马种类分析错误，且显示发现64个隐藏进程，但无法给出更详细的隐藏进程相关信息，取证效果较差。

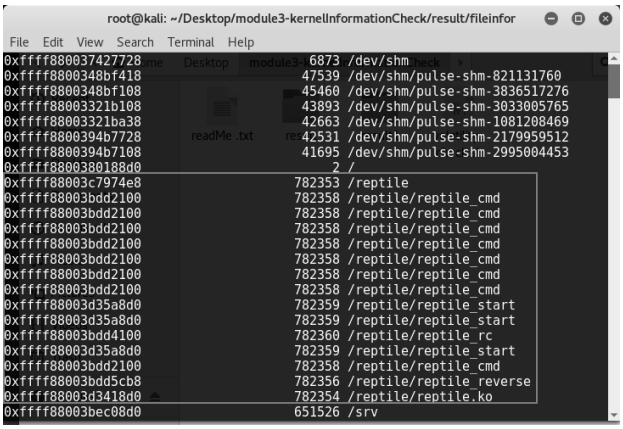
本文方法检测结果如图8所示。



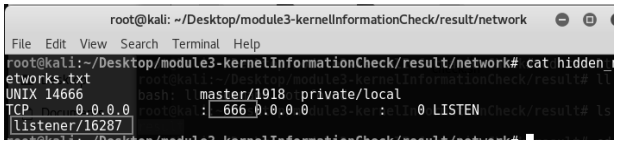
a）隐藏端口检测结果



b）隐藏内核模块信息



c）隐藏文件检测结果



d）隐藏网络连接信息

图 8 本文方法内核层 Rootkit 检测结果

由图8可以看出，本文方法能够检测出Reptile的隐藏端口、隐藏内核模块、隐藏文件及隐藏网络连接等关键信息，即可以检测出该木马已知部署的全部隐藏信息，这证明了本文方法取证效果显著。

5 结束语

本文针对Linux Rootkit恶意代码的特点及取证难点，提出一种基于Rootkit隐藏行为特征的Linux恶意代码取证方法，并利用近年来活跃的木马对取证方法进行实验验证，发现本文方法具有很好的检出效果和取证效果。对比实验表明，本文方法的检出效果及准确性均优于传统检测工具，可以对未知恶意代码进行取证。此外，本文方法具有自动化程度高、取证范围广、可扩展性好的特点，具备一定的现实意义与应用价值。（责编 马珂）

参考文献：

[1] DAFOULAS G A, NEILSON D, HARA S. State of the Art in Computer Forensic Education–A Review of Computer Forensic Programmes in the UK, Europe and US[C]//IEEE. 2017 International Conference on New Trends in Computing Sciences (ICTCS), October 11–13, 2017, Amman, Jordan. NJ: IEEE, 2017: 144–154.

[2] SI Tianhao, ZHANG Yi, LUO Yuqing. Overview of Linux Rootkit Implementation and Detection Technology[C]// CCF. ICoC. July 10, 2013, Zhangjiajie, China. Beijing: CCF, 2013: 123–128.
斯添浩，张怡，罗雨青 . Linux Rootkit 实现与检测技术综述 [C]// 中国计算机学会 . 中国互联网学术年会，2013–7–10，张家界，中国 . 北京：中国计算机学会，2013：123–128.

[3] LILLIS D, BECKER B, O'SULLIVAN T, et al. Current Challenges and Future Research Areas for Digital Forensic Investigation[EB/OL]. https://www.researchgate.net/publication/292996779_Current_Challenges_and_Future_Research_Areas_for_Digital_Forensic_Investigation, 2020–4–19.

[4] LIU Tao. Research and Implementation of Evidence Collection Based on Linux System[D]. Chengdu: University of Electronic Science and Technology of China, 2011.
刘涛 . 基于 Linux 系统的证据收集研究与实现 [D]. 成都：电子科技大学，2011.

[5] QIAN Zhigao. Research on Forensics Methods in Computer Intrusion Events[J]. Network Security Technology & Application, 2020, 20(4): 152–153.
钱志高 . 计算机入侵事件中的取证方法探究 [J]. 网络安全技术与应用，2020，20（4）：152–153.

[6] LIU Xiaoli, XU Bohe. Method and Practice of Network Intrusion Forensics Based on Linux System[J]. Network Security Technology & Application, 2016, 16(10): 34–35.
刘晓丽，徐博赫 . 基于 Linux 系统的网络入侵取证方法与实践 [J]. 网络安全技术与应用，2016，16（10）：34–35.

[7] HIRANO M, KOBAYASHI R. Machine Learning Based Ransomware Detection Using Storage Access Patterns Obtained From Live–forensic Hypervisor[C]//IEEE. 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), October 22–25, 2019, Granada, Spain. NJ: IEEE, 2019: 1–6.

[8] YARI I A, ZARGARI S. An Overview and Computer Forensic Challenges in Image Steganography[C]//IEEE. 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), June 21–23, 2017, Exeter, UK. NJ: IEEE, 2017: 360–364.

[9] LI Tonghong, WANG Ning, LIU Zhijun. Data Hiding and Anti–forensic on Modern Computer System[J]. Computer Systems & Applications, 2013, 22(5): 1–4, 37.
李佟鸿，王宁，刘志军 . 计算机系统信息隐藏反取证技术 [J]. 计算机系统应用，2013，22（5）：1–4，37.

[10] AL FAHDI M, CLARKE N L, LI Fudong, et al. A Suspect–oriented Intelligent and Automated Computer Forensic Analysis[J]. Digital Investigation, 2016, 18(9): 65–76.

[11] TORASKAR T, BHANGALE U, PATIL S, et al. Efficient Computer Forensic Analysis Using Machine Learning Approaches[C]//IEEE. 2019 IEEE Bombay Section Signature Conference (IBSSC), July 26–28, 2019, Mumbai, India. NJ: IEEE, 2019: 1–5.

[12] ZHANG Yu, LIU Qingzhong, LI Tao, et al. Research and Development of Rootkit[J]. Journal of University of Electronic Science and Technology of China, 2014, 44(4): 563–578.
张瑜，刘庆中，李涛，等 . Rootkit 研究综述 [J]. 电子科技大学学报，2014，44（4）：563–578.

[13] LIANG Shengrong, FAN Mingyu, WANG Guangwei, et al. New Method of Detecting Kernel-level Rootkit[J]. Application Research of Computers, 2009, 26(8): 3047-3049.
梁升荣，范明钰，王光卫，等 . 一种新的内核级 Rootkit 的检测方法 [J]. 计算机应用研究，2009，26（8）：3047-3049.
[14] YAN Renzhong, ZHONG Xichang, ZHANG Ni. A Method to Automatically Detect and Recover from Kernel Level Rootkit[J]. Computer Engineering, 2006, 32(10): 77-79.
颜仁仲，钟锡昌，张倪 . 一种自动检测内核级 Rootkit 并恢复系统的方法 [J]. 计算机工程，2006，32（10）：77-79.
[15] MUSAVI S A, KHARRAZI M. Back to Static Analysis for Kernel-level Rootkit Detection[J]. IEEE Transactions on Information Forensics and Security, 2014, 9(9): 1465-1476.
[16] CHEN Yuanpeng, LI Yongzhong. Analysis and Detect of Kernel-level Rootkit in Linux Platform[J]. Electronic Design Engineering, 2017, 25(1): 39-42.
陈远鹏，李永忠 . Linux 平台下 Rootkit 木马分析与检测 [J]. 电子设计工程，2017，25（1）：39-42.
[17] FANG Haifeng, ZHAO Yiqiang, ZANG Hongyong, et al. VMGuard: An Integrity Monitoring System for Management Virtual Machines[C]// IEEE. 2010 IEEE International Conference on Parallel & Distributed Systems, December 8-10, 2010, Shanghai, China. NJ: IEEE, 2011: 67-74.
[18] BEHROZINIA S, AZMI R. KLrtD: Kernel Level Rootkit Detection[C]//IEEE. 2014 22nd Iranian Conference on Electrical Engineering (ICEE), May 20-22, 2014, Tehran, Iran. NJ: IEEE, 2014: 1058-1063.
[19] LUCKETT P, MCDONALD J T, DAWSON J. Neural Network Analysis of System Call Timing for Rootkit Detection[C]//IEEE. 2016 Cybersecurity Symposium (CYBERSEC), April 18-20, 2016, Coeur d'Alene, ID, USA. NJ: IEEE, 2016: 1-6.
[20] LOBO D, WATTERS P, WU Xinwen. Identifying Rootkit Infections Using Data Mining[C]//IEEE. 2010 International Conference on Information Science and Applications, April 21-23, 2010, Seoul, South Korea. NJ: IEEE, 2010: 1-7.
[21] JOY J, JOHN A. A Host Based Kernel Level Rootkit Detection Mechanism Using Clustering Technique[M]//Springer. Trends in Computer Science, Engineering and Information Technology. Heidelberg: Springer, Berlin, Heidelberg, 2011: 564-570.
[22] FENG Peijun, ZHANG Ping, CHEN Zhifeng, et al. Design and Implementation of a New Linux Kernel-Level Rootkit[J]. Journal of Information Engineering University, 2016, 17(2): 231-237.
冯培钧，张平，陈志锋，等 . 一种新型 Linux 内核级 Rootkit 设计与实现 [J]. 信息工程大学学报，2016，17（2）：231-237.
[23] WANG Xueyang, KARRI R. Reusing Hardware Performance Counters to Detect and Identify Kernel Control-Flow Modifying Rootkits[J]. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 2016, 35(3): 485-498.
[24] ZHOU Liwei, MAKRIS Y. Hardware-assisted Rootkit Detection via On-line Statistical Fingerprinting of Process Execution[C]//IEEE. 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), March 19-23, 2018, Dresden, Germany. NJ: IEEE, 2018: 1580-1585.
[25] PETRONI N L, FRASER T, MOLINA J, et al. Copilot-a coprocessor-based Kernel Runtime Integrity Monitor[C]//USENIX. The 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA. Berkeley: USENIX Association, 2004: 13.
[26] AKAO Y, YAMAUCHI T. KRGuard: Kernel Rootkits Detection Method by Monitoring Branches Using Hardware Features[C]//IEEE. 2016 International Conference on Information Science and Security (ICISS), December 19-22, 2016, Pattaya, Thailand. NJ: IEEE, 2016: 1-5.
[27] MOON H, LEE H, HEO I, et al. Detecting and Preventing Kernel Rootkit Attacks with Bus Snooping[J]. IEEE Transactions on Dependable and Secure Computing, 2015, 14(2): 145-157.
[28] CARRIER B D, GRAND J. A Hardware-based Memory Acquisition Procedure for Digital Investigations[J]. Journal of Digital Investigation, 2004, 1(1): 50-60.
[29] BBN Technologies. Fred: Forensic Ram Extraction Device[EB/OL]. http://www. ir. bbn. com/vkawadia/, 2012-12-10.