系统服务 Rootkits 隐藏行为分析*)

龙 海 郝东白 黄 皓

(南京大学计算机科学与技术系 软件新技术国家重点实验室 南京 210093)

摘 要 用挂钩系统服务来实现进程、文件、注册表、端口等对象的隐藏是最常见的 rootkits 实现方式。然而大量的检测方法并不能将 rootkits 和其所隐藏的对象对应起来。本文分析了用户层和内核层系统服务 rootkits 的隐藏行为，建立了 6 种模型。在检测出系统服务 rootkits 的基础上，提出了一种分析其二进制执行代码，匹配模型，找出隐藏对象的方法，实现了一个隐藏行为分析原型。实验结果证明这种隐藏行为分析方法能有效分析出隐藏对象。

关键词 rootkits，系统服务，行为，控制流图，数据流图，函数调用图

Analyse the Undetectable Behavior of Rootkits on System Services

LONG Hai HAO Dong-bai HUANG Hao

(State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

Abstract Hooking the system services to hide the presence of objects such as processes, files, registry keys, and open ports is the most popular method of rootkits. But a great deal of rootkits detection methods can't tell the relationship between the rootkits and the hided objects. Analyzing the undetectable behavior of user-mode or kernel-mode rootkits on system services, six hide models are built. We develop a method to reveal the object hided by the rootkits when the rootkits are detected through analyzing the binary code of the rootkits function and matching the hide models. We implement a prototype of the method. The results of experiments on some famous rootkits demonstrate the effectiveness of the model.

Keywords Rootkits, System service, Behavior, Control-flow graph, Data-flow graph, Call graph

1 引言

Microsoft 的报告称在 Windows XP Service Pack 2 操作系统下有超过 20%的恶意代码使用了 rootkits 技术[1]。安全专业厂商 F-Secure 于 2006 年 12 月发表的一篇研究报告表明，rootkits 在过去的 6 个月里已经成为最大的安全挑战之一[2]。Rootkits 是入侵者在入侵目标系统后，隐藏行为和开辟后门的技术或者是工具；rootkits 可以让攻击者达到以下目的：隐藏恶意程序的进程，文件，注册表，端口；使用 rootkits 技术使得攻击程序如蠕虫、木马等恶意程序能逃避管理员和杀毒软件的检测[3,5,6,8,9]。从定义中可以看出，rootkits 最重要的功能是隐藏。

Rootkits 中针对系统服务占据主要部分(在本文中将称之为系统服务 rootkits)，无论是进程隐藏、文件隐藏、端口隐藏，还是注册表隐藏都能利用系统服务 rootkits 实现。Windows 操作系统下一些著名的 rootkits 程序如 Hacker defender，AFXRootkit2005，NTRootkit，H4eHook[4] 都使用了系统服务 rootkits 技术。

本文提出了一种找出系统服务 rootkits 的隐藏对象的方法，其思想是：首先对系统服务 rootkits 进行分析，建立系统服务 rootkits 实现隐藏的模型。在 rootkits 检测出来的基础上，利用反汇编工具 IDA32 对二进制代码进行反汇编分析，得到 rootkits 函数的控制流图、函数调用图和数据流图、与系统服务 rootkits 模型进行比较，找到关键的比较函数分析出系统服务 rootkits 的隐藏对象。本文的实现环境是在 Windows 平台下。

2 相关工作

现在关于 rootkits 的研究主要集中在 rootkits 的检测，检测方法按检测的对象可以分为：检测 rootkits 本身；检测隐藏对象。

检测 rootkits 本身可以分为特征码检测(signature)、完整性检测(integrity)，以及利用 rootkits 特征的静态检测。杀毒软件使用得比较多的是特征码检测。完整性检测主要是比较磁盘文件与内存中文件的差别来判断是否存在 rootkits，一些检测工具对系统服务 rootkits 的检测使用这种方法，比如 SystemVirginity Verifier(SVV)，Icesword，VICE。文献[5,6]根据内核 rootkits 需要可加载内核模块和对特定内存进行修改的特征，提出在加载内核模块时，对二进制文件进行分析的检测方法。

检测隐藏对象主要是不同视角的检测(Cross-view Comparisons)，主要从不同角度来查看系统的状态信息，从而检测出系统中被隐藏的对象，Strider Ghostbuster[7] 检测注册表和文件，文献[8]中检测用户层 rootkits 隐藏对象均采用不同视角检测隐藏对象的方法。

检测 rootkits 本身并不能知道 rootkits 所隐藏的对象，而

*)863 项目：分布式可信计算系统研究(2007AA1Z409)。龙 海 硕士，主要研究领域为网络安全；郝东白 硕士，讲师，研究方向为网络安全；黄 皓 教授，博士生导师，主要研究领域为网络信息安全。

检测隐藏对象则不知道是由什么 rootkits 来实现隐藏的。本文的意义在于:将这两种检测方法联系起来,先检测出系统服务 rootkits 本身,分析 rootkits 的隐藏行为,找到 rootkits 所隐藏的对象,进而分析恶意软件各个模块的关系,得到更为完整的恶意软件样本。

3 系统服务 rootkits 隐藏行为分析

3.1 系统服务 rootkits

系统服务是操作系统核心向用户提供操作硬件设备、请求内核服务的接口。系统调用接口位于用户态与核心态之间的边界,用户程序通过系统调用,向操作系统内核请求服务,操作系统内核完成服务后将结果返回给用户进程,它提供了操作系统环境由用户态切换到内核态的功能。Windows 系统服务的调用路径如图 1 所示。在文献[3]中比较详细地介绍了几种系统服务 rootkits 的方式:用户模式 Hook,SSDT 表重定向,SSDT 表项修改,服务函数重定向。

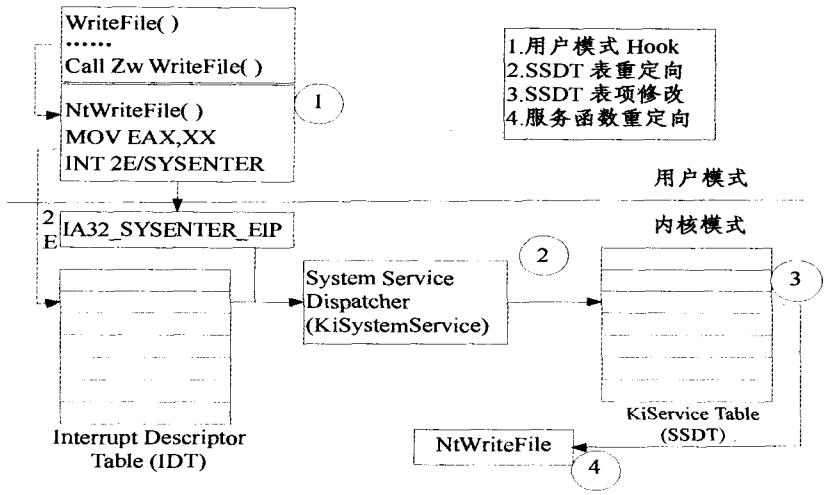

图 1 系统服务 rootkits

一些知名 rootkis 程序所使用的系统服务 rootkits 技术[4,10],如表 1 所示。

表 1 一些知名 rootkits 使用的系统服务 rootkits 技术

Rootkit 名称	Rootkit 类型	检测出挂钩系统服务数目
Hacker defender	用户模式 hook	14
AFXRootkit2005	用户模式 hook	5
NTRootkit	SSDT 表项修改	21
H4eHook	SSDT 表项修改	3
Backdoor-CWX	SSDT 表项修改	3

3.2 隐藏行为分析

分析系统服务 rootkits 的隐藏行为主要是根据系统服务 rootkits 的特点,找出其隐藏的对象。Rootkits 实现隐藏的方式分为两类[8]:一类是对用户层或者是内核层的资源枚举函数的结果进行过滤,第二类是把隐藏对象从操作系统的相关链表中删除。系统服务 rootkits 属于前者。利用系统服务实现隐藏主要是在系统服务调用后返回给应用程序一个过滤后的结果。

在文献[9]中提到一个 rootkits 的实现模型:

一个理想的 rootkit 程序的目标是替换目标机器上一个已经存在的程序,原来程序的功能必须看起来保持不变,同时增加一些额外的带有恶意的功能,可以描述为如下公式:

RootkitFunction=OriginalFunction⊕AdditionalFunction

具体应用到系统服务 rootkits:RootkitFunction 是系统服务 Rootkits 函数;OriginalFunction 是系统服务本身,在 rootkits 实现中,直接调用原来的函数;AdditionalFunction 则是附加的功能,是对返回结果进行过滤实现隐藏的功能。AdditionalFunction 只有在返回结果与隐藏对象有关的情况下才进行过滤。将这个模型在系统服务 rootkits 上推广,分析 AdditionalFunction 是如何判断返回结果是否与隐藏对象有关,以及如何对结果进行过滤的。

表 2 系统服务 rootkits 模型

Model(1,1): if condition (Parameters) = TRUE return IllegalOperation; else return OriginalFunction (Parameters);	Model (2,1): result = OriginalFunction (Parameters); if condition(result)=TRUE return IllegalOperation; else return result;
Model (1,2): if condition (Parameters) = TRUE return OriginalFunction (OtherObject); else return OriginalFunction (Parameters);	Model (2,2): result = OriginalFunction (Parameters); if condition(result)=TRUE return OriginalFunction (OtherObject);; else return result;

Model (1,3):
result = OriginalFunction (Parameters);
if condition (result)=TRUE{
Delete Hide-Object from result;
return result;
}
return result;
	Model (2,3):
	result = OriginalFunction (Parameters);
	if condition (result)=TRUE{
	Delete Hide-Object from result;
	return result;
	}
	return result;

判断返回结果是否与隐藏对象有关可以根据当前参数提前判断,也可以根据 OriginalFunction 的返回结果进行判断,因此有 2 种判断条件:

1. 可以直接从参数判断返回结果是否与隐藏对象有关。

2. 不能从参数判断返回结果是否与隐藏对象有关,需要从 OriginalFunction 返回结果判断。

按照文献[9]中的模型,为了不影响程序的其它功能,AdditionalFunction 对返回结果进行过滤分为 3 种过滤方法:

1. 对返回结果只与隐藏对象相关的,返回一个非法操作的结果。

2. 对返回结果只与隐藏对象相关的,返回一个其他对象的值。

3. 对返回结果包含隐藏对象的，在结果中删除隐藏对象，将其他结果返回。

将判断条件和过滤方法组合起来将得到系统服务 rootkits 可能的 6 种模型，Model(x,y)，x 表示判断条件，y 表示过滤方法。

3.3 基于流图分析寻找隐藏对象

在上一节我们已经分析了系统服务 rootkits 可能的 6 种模型，对一个 rootkit 函数，如果能分析出与这六种模型中的哪一种匹配，在控制流上就能找到判断条件 condition（result）。在判断条件中需要从参数或者 OriginalFunction 的结果来判断是否与隐藏对象相关，一般是从参数或者返回结果得到当前操作对象的特征与隐藏对象特征的比较，找到关键的比较函数，分析其参数，就能找到隐藏对象。

首先分析系统服务 rootkits 函数的控制流图，基本的控制流图的节点，入口由其它节点跳入，在节点最后一句语句发生跳转，在节点内部不存在任何分支和中断（halt）。在我们的实现中对控制流图做了简化，控制流图只包含三种节点：1）调用 OriginalFunction 的节点保持不变；2）对返回结果进行操作的节点保持不变；3）其他节点按执行次序进行合并。简化后的控制流图就能与上一小节中的 6 种模型进行匹配，分析出 condition()函数在控制流图的哪一个节点中。

为了判断返回结果是否与隐藏对象有关，在 condition()中存在关键的比较函数，与隐藏对象特征（一般为隐藏对象的名字，比如进程名，注册表键值名称，文件名）的比较。分析 rootkits 函数的函数调用流图，根据具体的模型找到导致控制流图分支的比较函数就是关键比较函数。

分析系统服务 rootkits 函数的数据流图，是由上一节判断条件所描述的，与参数或者 OriginalFunction 返回结果无关的便是隐藏对象的特征。

图 2 是一个具体的分析实例。

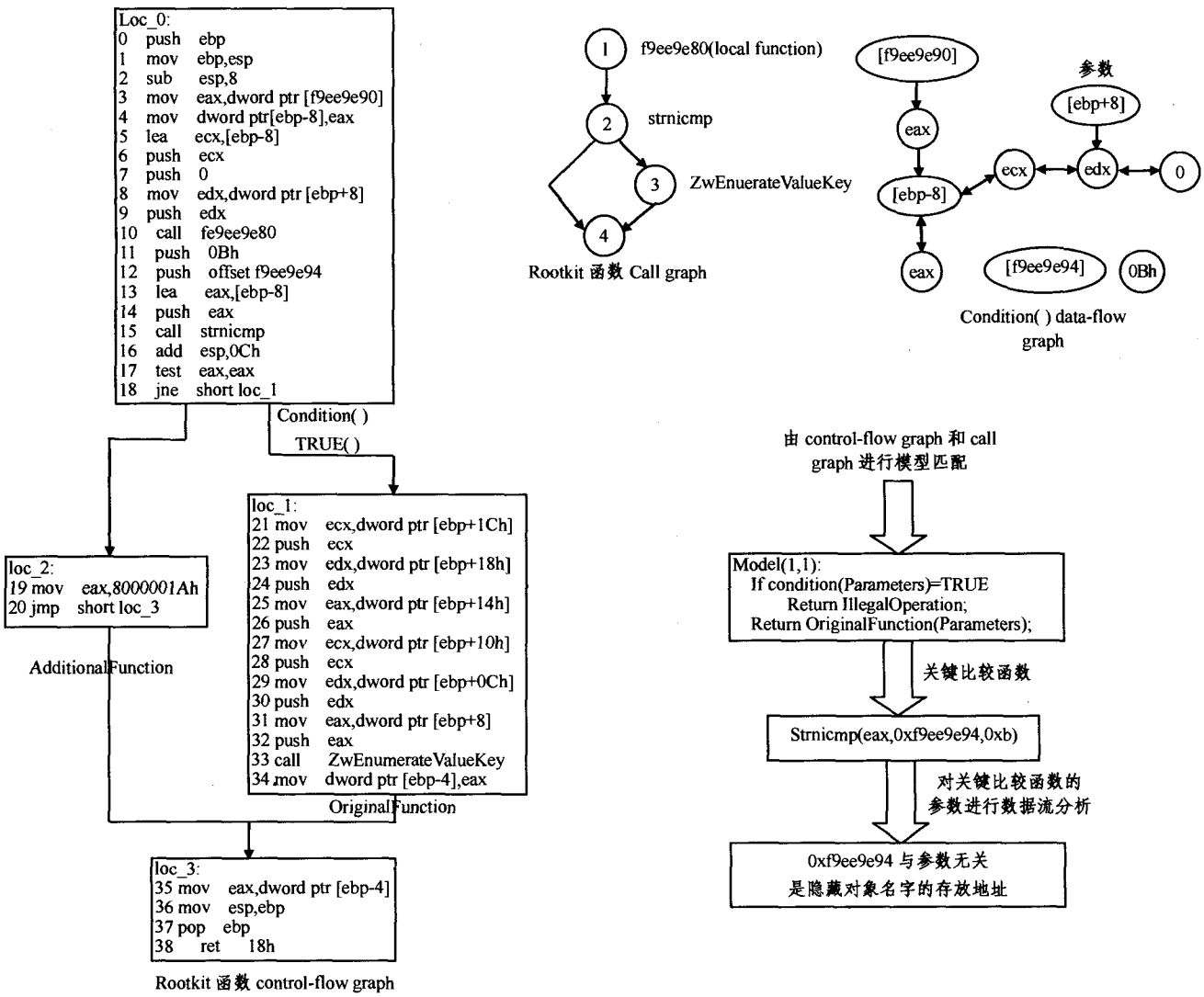

图 2 一个隐藏行为分析的实例

在地址 0xf9ed0a28 处的一个 ZwEnumerateValueKey 系统服务的 rootkit，找到这个 rootkit 函数的结尾，通过 IDA32 对这段指令代码进行分析，得到简化的控制流图和函数调用流图，与模型进行对比是 Model(1,1)，找到 condition()函数所在块，结合函数调用图，找到关键的比较函数 strnicmp。其参数为(eax,0xf9ee9e94,0xb)，对其参数进行数据流分析，知道 eax 与参数[ebp+8]有关，与 0xf9ee9e94 无关，分析 0xf9ee9e94 地址处的内容，得到隐藏对象的名字。

4 实现

4.1 检测系统服务 rootkits

检测系统服务 rootkits 我们采用相关工作中描述的完整性检测方法。

用户模式 hook 实现的系统服务 rootkits(图 1)，有两种

实现方式:第一种是对 ntdll. dll 的导出函数地址进行修改,第二种实现方式是采用 hook(detour)的方法,即修改函数的头几个字节为一条 jump 语句。对第一种方式我们采取的方式是在内存中对 ntdll. dll 的导出函数进行扫描,如果函数地址不在 ntdll. dll 模块内,则认为是 rootkits 函数地址。对第二种方式则是扫描内存中 ntdll. dll 的头几个字节,如果发现 jmp 语句,则认为目的地址是 rootkits 函数地址。

内核模式的三种实现方式(图 1),我们采用硬盘文件完整性检测方法[11]。加载一个内核驱动,读取内核变量的 KiServiceTable 地址,比较磁盘文件(ntoskrnl. exe)上 KiServiceTable 的相对地址检测 SSDT 表重定向,比较磁盘文件上和内存中表项的相对地址,检测 SSDT 表项修改,扫描内存中系统服务函数的头几个字节检测系统服务函数重定向。

4.2 分析原型

我们根据第 3 节分析的 rootkits 隐藏行为分析理论,实现了一个隐藏行为分析的原型系统。检测到系统服务 rootkits 函数的地址后,以一种启发式的方法找出 rootkits 函数的结尾。函数一般以{push ebp;mov ebp,esp}开始,{mov esp,ebp;pop ebp;ret}结尾,从 rootkits 函数地址开始扫描,碰到一个函数结尾或者碰到一个函数开头则结束。从扫描地址开始到扫描结束地址是 rootkits 函数的二进制代码段。对找到的二进制指令段使用有编程接口的 IDA32 进行反汇编分析,在此基础上进行控制流分析、函数调用分析、数据流分析,匹配第 3 节中的模板,找到 condition(FunctionEnviroment)和 AdditionalFunction;然后在其中找到关键的比较函数,分析函数参数,找到隐藏对象。实现框架如图 3 描述。

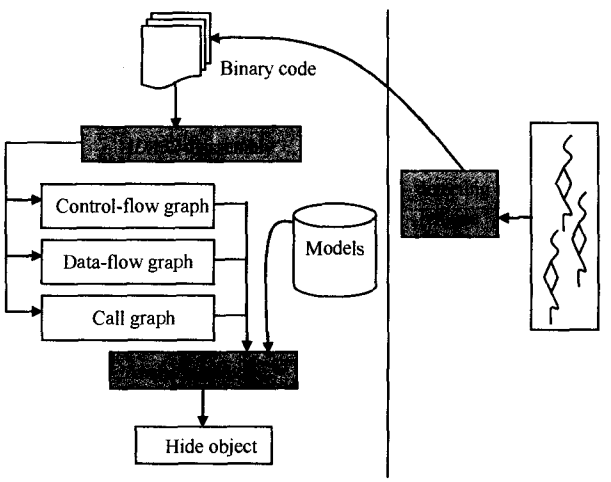

图 3 隐藏行为分析框架

5 实验结果

我们将模型应用到一些著名的 rootkits[4,10]进行检测和分析,得到实验结果。

实验结果表明无论是对于内核系统服务 rootkits 还是用户层系统服务 rootkits,原型系统都是有效的,并且取得了较高的分析精度。一些误报的情况主要集中在对关键的比较函数的寻找上,同时一些非常规的编程手段[12]也会增加分析原型系统的误报率。

表 3 系统服务 rootkits 的行为分析结果

Rootkit 名称	检测出挂钩系统服务数目	准确分析出隐藏对象的 rootkits 数目	正确率	误报率
Hacker defender	14	10	71.4%	28.6%
AFXRootkit2005	5	3	60%	40%
NTRootkit	21	15	71%	29%
H4eHook	3	2	66.6%	33.4%
Backdoor-CWX	3	2	66.6%	33.4%

结束语 本文根据系统服务 rootkits 隐藏行为的特点,提出其隐藏行为的 6 种模型,在此基础上实现一个系统服务 rootkits 隐藏行为分析原型系统,并且有较好的实验效果。各种 rootkits 的隐藏行为具有相似的特点,对分析其他 rootkits 的行为具有借鉴意义。同时,一些防病毒软件如 Norton,Kaspersky 以及一些主机监控软件都会使用与系统服务 rootkits 相似的 hook 技术,挂钩系统服务,对系统进行监控和审计。改进本文提出的方法提高分析的准确性和智能化,应用到更广的 rootkits 行为分析,对 rootkits 的行为和一些使用相似技术的反病毒软件和监控软件的行为进行比较和区分将是本文下一步的工作。

参 考 文 献

[1] http://www.eweek.com/article2/0,1895,1896605,00.asp [EB]. Mar. 2007
[2] http://www.f-secure.com/2006/2/[EB]. Mar. 2007
[3] Ries C. www.vigilantminds.com/files/inside_windows_rootkits.pdf [EB]. Mar. 2007
[4] http://www.rootkit.com [EB]. Mar. 2007
[5] Kruegel C, Robertson W, Vigna G. Detecting kernel-level rootkits through binary analysis [C]// Computer Security Applications Conference, 2004. 20th Annual. Dec. 2004: 91-100
[6] 易宇,金然. 基于符号执行的内核级 Rootkit 静态检测[J]. 计算机工程与设计, 2006,27(16):3064-3068
[7] Wang Yi-Min, Beck D, Vo B, et al. Detecting Stealth Software with Strider GhostBuster. Microsoft Research Technical Report MSR-TR-2005-25. February 21, 2005 (submitted to DSN-2005 on December 13, 2004)//Proc. Int. Conf. on Dependable Systems and Networks (DSN-DCCS). June 2005
[8] Wang Yi-Min, Beck D. Fast User-Mode Rootkit Scanner for the Enterprise// 19th Large Installation System Administration Conference. Dec. 2005:23-30
[9] Levine J, Grizzard J, Owen H. Detecting and Categorizing Kernel-Level Rootkits to Aid Future Detection. IEEE Security & Privacy, 2006,4(1):24-32
[10] http://vil.nai.com/vil/content/v_137858.htm [EB]. Mar. 2007
[11] http://www.rootkit.com/newsread_print.php?newsid=176 [EB]. Mar. 2007
[12] Linn C, Debray S. Obfuscation of executable code to improve resistance to static disassembly// Proceedings of the 10th ACM conference on Computer and communications security table of contents. 2003: 290-299