第 36 卷 第 8 期 计 算 机 工 程 2010 年 4 月
Vol.36 No.8 Computer Engineering April 2010
·安全技术· 文章编号：1000—3428(2010)08—0161—02 文献标识码：A 中图分类号：N945

Linux 下 VFS 层 Rootkit 技术研究

丁 滟，富弘毅，李宇卓
(国防科技大学计算机学院，长沙 410073)

摘 要：Linux 下 VFS 层 rootkit 隐藏层次深，查杀难度大。其典型应用 adore-ng 在实际使用时无法屏蔽卡巴斯基等实时监控软件，破坏隐蔽效果。针对该问题，运用系统调用修改和 VFS 写函数内容过滤 2 种方法，设计并实现了相应的改进方案。仿真实验结果表明，该方案易于实现、效果良好，可以有效提高 adore-ng 的隐蔽性能。

关键词：Linux 内核；信息安全；rootkit 技术；VFS 层

Research on VFS Layer Rootkit Technique in Linux

DING Yan, FU Hong-yi, LI Yu-zhuo
(School of Computer, National University of Defence Technology, Changsha 410073)

【Abstract】 The kernel rootkit at VFS layer hides deeply in Linux, and it is hard to be detected and killed. adore-ng is a typical rootkit application, but it can not survive some of the real-time monitoring programs, such as the Kaspersky Internet security. Aiming at this problem, the paper proposes two different solutions. One is by the modification of relevant system calls, the other is by filtering the content written by the VFS write call. Both these two approaches are easy to be implemented. Experimental results show the approaches are effective.

【Key words】 Linux kernel; information security; rootkit technique; VFS layer

1 概述

网络时代带来了全球化信息共享，同时也使得信息安全成为备受关注的问题。无论是信息攫取还是信息保护，在当今信息战环境下，只有掌握“攻”、“防”两方面技术，才能处于有利地位。rootkit 是在攻击者已经获取超级用户访问权限之后，隐藏自己踪迹和保留访问权限的工具，为攻击者提供“保护伞”。近年来，随着 Linux 系统应用范围日渐广泛，针对该系统的 rootkit 技术也迅速发展，已经从简单的用户级命令替换发展为在 Linux 内核中进行信息拦截，从系统调用替换到 VFS 层函数篡改，隐藏层次逐渐深入，隐蔽性越来越高。VFS 层 rootkit 在当今隐藏技术中隐藏层次最深，检测难度大。研究掌握 VFS 层 rootkit 技术对提高网络攻防能力，在信息战环境下立于不败之地具有重要意义。

本文对当前流行的 VFS 层 rootkit 技术以及经典应用实例 adore-ng 进行分析研究，针对该软件在杀毒软件实时监控下暴露的缺陷提出改进方案，并加以实现，以增强软件的隐蔽性。实验证明改进效果良好。

2 相关工作

rootkit 技术出现在 20 世纪 90 年代早期，主要用于黑客在侵入计算机之后，保持继续访问系统的能力。最初的 rootkit 只是后门程序，之后为了躲避检测工具，研究的方向转向于如何有效地隐藏攻击者的各种踪迹，包括访问的文件、启动的进程等[1]。

Linux 系统的 rootkit 技术发展经历了用户级 rootkit 和内核级 rootkit 2 个阶段[2]。用户级 rootkit 通过在用户层替换 ls, ps 等关键命令，屏蔽攻击者的文件、进程等信息；内核级 rootkit 则基于可装载内核模块(Loadable Kernel Modules, LKM)技术，将具有隐藏功能的模块嵌入系统内核，向用户屏蔽攻击者的入侵活动。

内核级 rootkit 又细分为系统调用表修改类、系统调用表重定向类以及 VFS 层 rootkit 等。系统调用表修改类 rootkit 通过修改导出的系统调用表，对与攻击行为相关的系统调用进行替换，隐藏攻击者的行踪，典型应用有 Knark, adore 等；系统调用表重定向类 rootkit 并没有修改系统调用跳转表的内容，而是首先拷贝了系统调用表，然后将拷贝的系统调用表按照入侵者的意图进行修改，执行入侵者改写的系统调用响应函数。然后将 system_call 从旧的系统调用表上移开，指向新的系统调用表。此类 rootkit 的典型应用有 SucKIT 等；VFS 层 rootkit 并不修改系统调用层的内容，而是通过修改 VFS 层的具体处理函数，如替换 VFS 层的 file_ops 等函数，来实现信息隐藏目的。此类 rootkit 的典型应用有 adore-ng。

前 2 类内核级 rootkit 可以通过检查/dev/kmem 来发现系统调用表是否被篡改或替换[3]。而对于 VFS 层 rootkit，此类工具无法正确检出，一些专门针对 VFS 层 rootkit 的检测工具基本处于研究阶段[4]，尚无通用工具出现。通常对于 VFS 层 rootkit 只能够通过完整性校验等工具发现可疑，然后进一步排查。

由上述可见，基于 VFS 层的 rootkit 具有隐藏层次深、查杀难度大等特点，成为当今 rootkit 研究的热点。对该类 rootkit 进行深入研究，有利于掌握最新的隐藏技术，提供网络攻击能力，为打赢信息战做好充分准备。另外，对攻击技术的深

基金项目：国家“863”计划基金资助项目“分布加密存储软件结构及其关键技术”(2007AA01Z408)

作者简介：丁 滟(1977－)，女，硕士，主研方向：操作系统安全，网络安全；富弘毅，博士研究生；李宇卓，工程师

收稿日期：2009-12-01 **E-mail：**dingyan_ding@yahoo.com.cn

入研究也可以提高人们进行信息安全保护的能力。

3 adore-ng 防实时监控研究

作为典型并广泛应用的 VFS 型 rootkit，adore-ng 通过截获 VFS 层的函数实现了文件目录隐藏、进程隐藏、通信连接隐藏、信息过滤等 rootkit 功能[5]。

由于 adore-ng 工作于 VFS 层，隐蔽层次深，因此一般 rootkit 检测工具很难发现。然而对于一些工作在内核层的实时监控软件，虽然不会对 adore-ng 的存在进行报警，但会记录下 adore-ng 所访问的文件以及进程活动等。

以目前流行的病毒查杀工具卡巴斯基为例，卡巴斯基实时监控模块采用系统调用替换的方法，对系统调用进行拦截，获取系统文件、进程等访问行为。如图 1 所示，卡巴斯基替换了内核的 sys_open 函数，由于 adore-ng 通过修改 VFS 层 readdir 函数隐藏文件，并不对 sys_open 函数做处理，因此隐蔽进程访问隐蔽文件等操作就会被卡巴斯基实施监控模块记录下来。

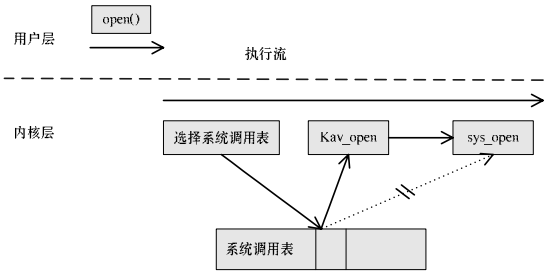

图 1　卡巴斯基实时监控过程

管理员检查实时监控日志，就可能发现被隐藏的文件、进程等，通过对这些信息的分析，就可能使攻击者的来源、目的全盘暴露。因此，有必要对 adore-ng 进行改进，使其适应实时监控环境，增强隐蔽性。

3.1 改进方案 1

通过上述分析可知，卡巴斯基通过修改系统调用来记录用户的行为。那么，如果截取修改过的系统调用，对其所记录的信息进行过滤即可实现屏蔽敏感信息。在此，仅以 open 系统调用为例，其余的系统调用截获类似。

如图 2 所示，在此方案中，首先查找出系统中原有的 sys_open 实现函数的地址，然后在此基础上设计实现自己的 open 调用实现函数 our_open。该函数首先判断所要 open 的文件是否是需要隐藏的文件，若是，则调用系统中原有的 sys_open 函数；否则调用 Kav_open，让卡巴斯基实时监控函数正常审计。最后修改系统调用表，使其指向 open 实现函数。

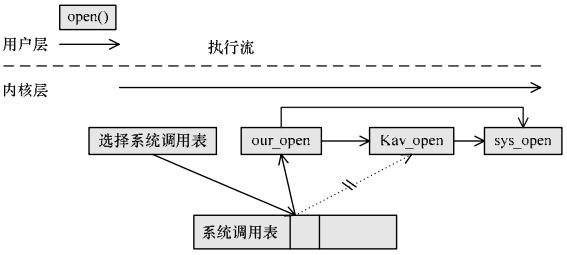

图 2　改进方案 1

通过上述修改，使得对被 adore-ng 隐藏的文件访问不会被记录下来，安全管理员无法通过分析实时监控日志而发现这些文件的存在；而对系统中其他文件的访问则会被正常记录，安全管理员不会察觉实时监控功能已经受到屏蔽。

3.2 改进方案 2

为了使 rootkit 能够绕过卡巴斯基的日志监控系统，可以考虑在 VFS 层对写入卡巴斯基日志文件的内容进行过滤。这就需要截获 VFS 层 file_operations 的 write 函数，使用笔者自己实现的 wirte 函数进行替换。函数具体流程设计如下：

(1)判断写操作的对象是否为卡巴斯基日志文件，若是，则继续(2)；否则跳转到(4)；

(2)判断当前进程是否为 adore-ng 所隐藏的进程，若是，则返回；否则继续(3)；

(3)判断当前缩写内容是否有 adore-ng 所隐藏的文件或目录名，若是，则返回；否则继续(4)；

(4)调用系统中原有 write 函数。

通过上述修改，如果此次写操作是向卡巴斯基日志文件中记录被隐藏文件的访问，那么此次写操作为空操作。而其他写操作则被正常记录。

3.3 改进结果

根据上述设计方案，分别对 adore-ng 进行修改，并且在真实环境对改进后的软件进行测试。

测试平台为 Fedora Core 5，内核版本 2.4.20，adore-ng 的版本为 1.54。测试流程如下：

(1)开启卡巴斯基实时监控系统；

(2)加载改进后的 adore-ng 模块，隐藏/tmp/test.c 文件；

(3)打开/tmp/test.c 文件；

(4)查看卡巴斯基日志文件，查找有关/tmp/test.c 的记录；

(5)卸载改进后的 adore-ng 模块，加载未经改进的 adore-ng 模块，并隐藏/tmp/test.c 文件；

(6)打开/tmp/test.c 文件；

(7)查看卡巴斯基日志文件，查找有关/tmp/test.c 的记录。

改进方案 1 及改进方案 2 的的测试结果分别如图 3、图 4 所示。

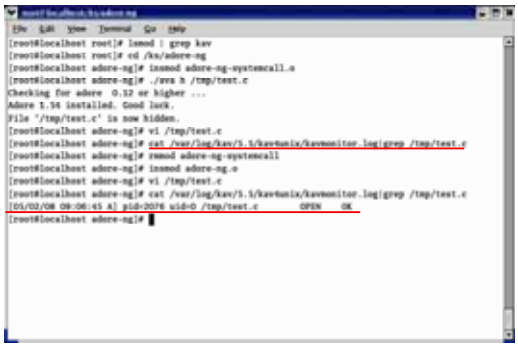

图 3　改进方案 1 测试结果

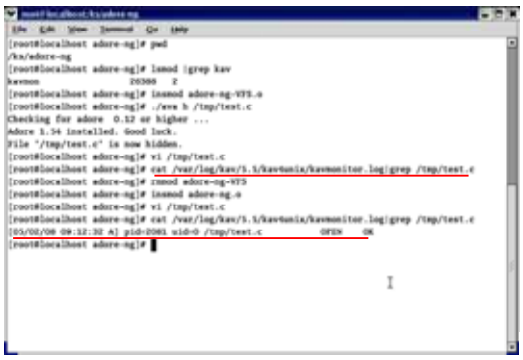

图 4　改进方案 2 测试结果

(下转第 164 页)