多款恶意软件的数据特征,由于已提取的特征是多款恶意软件运行过程中数据特征的子集,根据检测算法 JudgeMalware 可知已提取的特征可检测出恶意软件.

为了检验本文方法的误报率,我们使用了一些常用应用程序进行对比检测,其中,LTP(linux test project)测试工具集是目前较为流行的 Linux 基本功能测试集,它包含了多个子功能测试模块,例如系统调用、系统命令、内存分配、磁盘读写、文件系统、网络、数学运算测试等,对 Linux 内核进行各项长时间(24 小时)的测试,可较充分地覆盖内核代码.同时,在运行这些测试程序时给予不同的输入数据或者配置,使其能够运行更多的功能.根据提取的恶意软件数据特征对其进行检测,通过特征匹配判断是否会造成误报,结果见表 6.

Table 6 Results of false positive test

表 6 误报测试结果

benchmark	命令	静态数据类			动态数据类			误报
		数据类型	写位置	写次数	数据类型	写位置	写次数	
内核编译	make	30 450	14 568	165 912	565	6 989	79 358	×
解压缩 bzip2	tar –zxf linux-source.tar.bz2	30 450	8 657	109 247	565	6 780	74 985	×
压缩 gzip	tar –zcf linux-source-dir	30 450	8 631	99 879	565	6 693	75 134	×
unixbench 5.1.2	./Run	30 450	16 515	191 531	563	8 484	104 373	×
Linux 启动	linux boot	30 450	8 515	101 562	564	6 784	74 657	×
LTP	./runalltesh.sh	30 450	31 785	265 714	1128	15 879	210 367	×
apache eberver	./apachectl start	30 450	15 253	180 267	570	7 127	112 589	×
mysql	Service mysql start	30 450	16 308	188 674	565	6 954	83 521	×
thttpd	thttpd thttpd.conf	30 450	17 469	210 679	583	8 739	142 175	×

由表 6 可知,所有的 benchmark 均没有误报产生.这也进一步验证了提取的数据特征是合理的、有效的.

为了比较测试结果,我们使用现有的内核恶意软件检测工具与本文的原型系统进行对比分析,对比结果见表 7.

Table 7 Results of comparison test

表 7 对比测试结果

rootkit	攻击类型	secVisor	Gibraltar	NumChecker	MDS-DCB (ours)
adore 0.38	代码植入、控制数据篡改	√	√	√	√
sk 1.3b	代码植入、控制数据篡改	×	×	√	√
enyelkm v1.2	代码植入和篡改	×	×	√	√
override	代码植入、控制数据篡改	√	√	√	√
wipemod	数据篡改	√	√	×	√
kbeast v1.0	代码植入、控制数据篡改	√	√	√	√
synapsys	代码植入、控制数据篡改	√	√	√	√
CaRoGNa	代码植入、控制数据篡改	√	√	√	√
JOP_hideprocess	代码复用、非控制数据篡改	×	√	×	√
JOP_Synapsys	代码复用、控制数据篡改	×	√	√	√
ptrace_detach_hook[15]	非控制数据篡改	×	×	×	√

根据表 7 可以看出:

- secVisor 无法检测基于 JOP,ROP 实现的 rootkit 和 ptrace_detach_hook,这是因为 JOP_hideprocess,JOP_Synapsys 和 ptrace_detach_hook 绕过了代码完整性的缘故;
- Gibraltar 无法检测 sk 1.3b,enyelkm 和 ptrace_detach_hook,这是因为这 3 款 rootkit 未违背 Gibraltar 提出的数据不变量属性;其能够检测基于代码复用攻击的 JOP_hideprocess 和 JOP_Synapsys,是因为这两款 rootkit 违背了 $run\_list \subseteq all\_tasks$ 和系统调用表表项的值被破坏;
- NumChecker 无法检测 wipemod,JOP_hideprocess 和 ptrace_detach_hook,这是因为 NumChecker 是通过检测系统调用执行变化情况,而这 3 款 rootkit 并未修改任何系统调用;
- MDS-DCB 则检测出了所有的 rootkit,这也表明了本文提出的方法较现有的内核恶意软件检测工具具有更强的检测能力,能够检测代码植入、代码篡改、代码复用、控制数据篡改和非控制数据篡改等恶

意攻击.

5.3 性能测试

5.3.1 微基准测试

我们采用 lmbench 作为本次实验的微基准测试集.由于 MDS-DCB 基于内存分配/释放函数监控和内存读写监控分析技术实现,为了充分测试 MDS-DCB 的性能开销,我们从 lmbench 选择了与内存相关的系统调用、缺页处理和内存读写等作为基准测试指标.通过分别测试 MDS-DCB、Bitvisor、裸机三者下的性能开销,并通过对比来分析 MDS-DCB 所引入的性能开销,如图 9 所示.

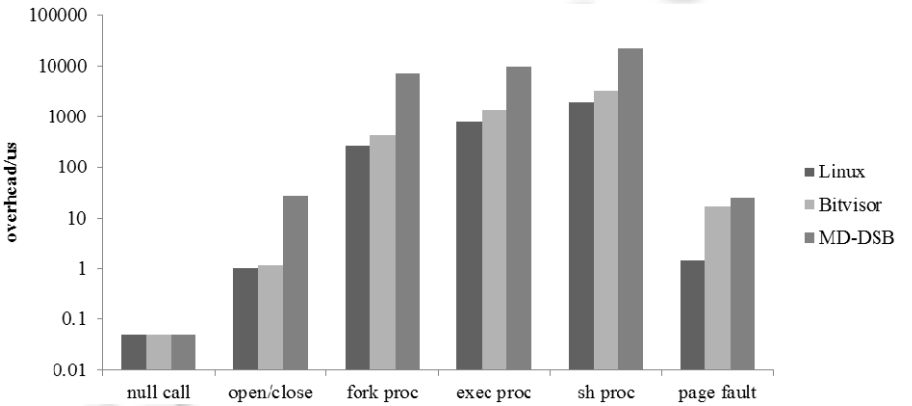

Fig.9 Microbenchmark results of MDS-DCB

图 9 MDS-DCB 微基准测试结果

其中,由于不同程序测试的性能开销数据差距较大,故我们对测试数据(纵坐标)进行对数化(基数为 10)处理.根据图 9 的数据可知:由于 MDS-DCB 未对 Null Call 操作进行监控,直接由处理器处理,故未带来任何开销;而 open/close 操作是先打开文档后关闭文档,这一过程中创建了 file 内核对象,MDS-DCB 对 file 结构的创建和使用进行了监控,Linux,Bitvisor 未进行监控,故 MDS-DCB 较 Linux 和 Bitvisor 带来了额外开销;同时,MDS-DCB 在 fork proc,exec proc 和 sh proc 这 3 项测试指标下的性能远大于其他指标,并且开销比 Bitvisor 大很多,这是因为这 3 项指标的测试过程涉及到进程类对象、文件类对象、内存类对象的创建、使用等操作,MDS-DCB 对这些内核对象进行了监控.prot fault 指的是保护异常带来的开销,MDS-DCB 和 Bitvisor 一样只是捕获该操作,捕获之后将该操作的处理返还给操作系统内核,故开销较小;page fault 指的是缺页异常带来的开销,由于 MDS-DCB 设置了监控的内核数据所在页的写属性,当产生写关键数据操作时,触发缺页异常,由 VMM 处理该异常,处理之后再返回内核,所以 page fault 所需要的开销较大,且大于 prot fault 的开销.

5.3.2 应用程序基准测试

为了进一步评测 MDS-DCB 的性能,我们选择了与误报测试一样的应用程序基准测试集测试 MDS-DCB 的性能开销.测试结果如图 10 所示.

由于内核编译和 unixbench 是综合性应用,既需要 CPU 时间,也需要大量的 I/O 操作,测试过程中大量使用系统资源如文件系统、管道和进程等,这些行为调用了内核服务如系统调用和缺页处理而触发内核的内存活动,而 MDS-DCB 监控内核对象的创建、读写等行为,故带来了较大的性能开销.对于解压缩和压缩程序,分别引入了 15%和 11%的性能开销,它们属于计算密集型应用,相对内核编译和 unixbench 开销较小.Linux 启动可全面测试 MDS-DCB 的性能,因为系统启动过程中囊括了 MDS-DCB 的所有监控和操作的内容,该项测试引入的开销为 45%.综上测试,MDS-DCB 的性能开销在一个可以接受的范围内.

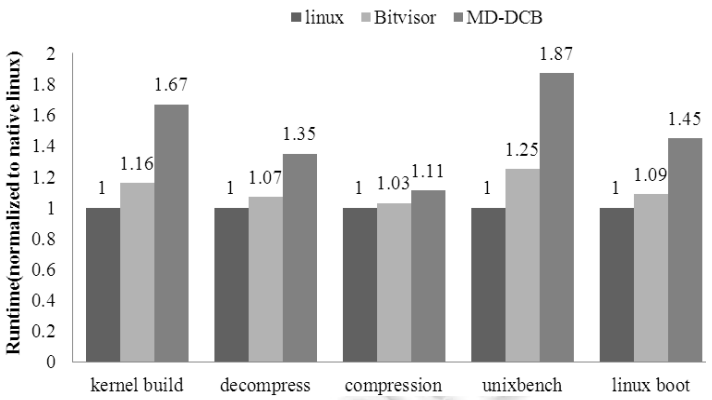

Fig.10 Application benchmark performance test results

图 10 应用基准程序性能测试结果

5.4 方法存在的不足

实验过程中选择的恶意代码测试用例都是目前公开的,由于特征库中存储有他们所有的特征值,所以只要特征值是合理有效的,那么均可以被检测出来.对于未知的内核恶意软件,若采用上述提出的特征匹配算法,那么将可能会产生漏报,因为在未知恶意软件运行过程中提取的数据特征集与已知特征无法匹配成功.另外,对于恶意软件修改自己行为迷惑检测软件的情况,本文的方法也有局限性.修改主要包括两种情况,如图 11 所示.

- 第 1 种情况,如图 11(a)所示,攻击者在不改变原恶意软件功能的基础上增加了一些冗余功能,那么假设新的恶意软件对应的数据特征为 S',过程中采集的数据访问行为集为 D,根据匹配算法,首先求解 D 与 S_M 的交集.从图 11 中可知,该运算结果为 S_M.所以,这种情况下仍可以检测出恶意软件;
- 第 2 种情况,如图 11(b)所示,攻击者修改了原恶意软件的部分功能并增加一些冗余功能,那么同样假设新的恶意软件对应的数据特征为 S',过程中采集的数据访问行为集为 D,根据匹配算法,首先求解 D 与 S_M 的交集.从图中可知,该运算结果为 $S'-R$.而$(S'-R)\neq S_M$,根据算法可知,无法判断是否存在恶意软件,从而造成匹配失效,导致漏报.对于以上提到的不足,我们需要进一步研究不同恶意软件数据特征之间的关系,降低漏报率,提升检测准确率.

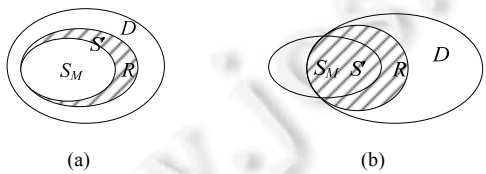

Fig.11 Data characteristic relation between malware and modified malware

图 11 恶意软件修改自身行为后的数据特征关系

6 结 论

本文提出了一种基于数据特征的内核恶意软件检测方法,并设计实现了原型系统 MDS-DCB.该方法基于内核对象访问模型给出了检测方法的基本思路;在内核运行过程中监控内核动态数据的分配事件和内核数据的访问构建恶意软件数据特征;并基于该数据特征检测内核恶意软件.MDS-DCB 在 VMM 层实现,无需修改操作系统内核,具有良好的兼容性和安全性.通过对原型系统的有效性和性能进行测试,测试结果表明:MDS-DCB 能够有效检测内核恶意软件,并且引入的性能开销在一个可接受的范围内.目前,未对分析得到的恶意软件特征

进行处理,特征匹配是通过对比每条 SKDAP 实现的,这将影响检测效率,下一步将研究基于 SKDAP 的特征值计算问题.同时,不同的内核恶意软件之间存在相同或者类似的数据特征,故,特征的相似性问题也是下一步需要研究的内容.

致谢 在此,向对本文研究工作提供基金支持的单位和评阅本文的审稿专家表示衷心的感谢,向为本文研究工作提供基础和研究平台的前辈致敬.

References:

[1] Seshadri A, Luk M, Qu N, Perrig A. SecVisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. In: Proc. of the 21st ACM SIGOPS Symp. on Operating Systems Principles. New York: ACM Press, 2007. 335−350. [doi: 10.1145/1294261.1294294]

[2] Riley R, Jiang X, Xu D. Guest-Transparent prevention of kernel Rootkits with VMM-based memory shadowing. In: Proc. of the 11th Int'l Symp. on Recent Advances in Intrusion Detection. Cambridge: Berlin, Heidelberg: Springer-Verlag, 2008. 1−20. [doi: 10.1007/ 978-3-540-87403-4_1]

[3] Hund R, Holz T, Freiling FC. Return-Oriented rootkits: Bypassing kernel code integrity protection mechanisms. In: Proc. of the 18th Conf. on USENIX Security Symp. Berkeley: Usenix, 2009. 383−398.

[4] Carlini N, Wagner D. Rop is still dangerous: Breaking modern defenses. In: Proc. of the 23rd Conf. on USENIX Security Symp. Berkeley: Usenix, 2014. 385−399.

[5] Bletsch T, Jiang X, Freeh VW, Liang Z. Jump-Oriented programming: A new class of code-reuse attack. In: Proc. of the 6th ACM Symp. on Information, Computer and Communications Security. New York: ACM Press, 2011. 30−40. [doi: 10.1145/1966913.1966919]

[6] Chen P. Research on the attack and defense techniques of code Reuse [Ph.D. Thesis]. Nanjing: Nanjing University, 2012 (in Chinese with English abstract).

[7] Kruegel C, Robertson W, Vigna G. Detecting kernel-level rootkits through binary analysis. In: Proc. of the 20th IEEE Annual Conf. on Computer Security Applications. Washington: IEEE CS Press, 2004. 91−100. [doi: 10.1109/CSAC.2004.19]

[8] Musavi SA, Kharrazi M. Back to static analysis for kernel-level rootkit detection. IEEE Trans. on Information Forensics and Security, 2014,9(9):1465−1476. [doi: 10.1109/TIFS.2014.2337256]

[9] Riley R, Jiang X, Xu D. Multi-Aspect profiling of kernel rootkit behavior. In: Proc. of the 4th ACM European Conf. on Computer Systems. New York: ACM Press, 2009. 47−60. [doi: 10.1145/1519065.1519072]

[10] Wang X, Karri R. NumChecker: Detecting kernel control-flow modifying rootkits by using hardware performance counters. In: Proc. of the 50th Annual Design Automation Conf. New York: ACM Press, 2013. 79−86. [doi: 10.1145/2463209.2488831]

[11] Sharif MI, Lanzi A, Giffin JT, Lee W. Impeding malware analysis using conditional code obfuscation. In: Proc. of the 16th Annual Network and Distributed System Symp. Washington: Internet Society, 2008. 65−88.

[12] Fan W, Lei X, An J. Obfuscated malicious code detection with path condition analysis. Journal of Networks, 2014,9(5):1208−1214. [doi: 10.4304/jnw.9.5.1208-1214]

[13] Li J, Wang Z, Bletsch T, Srinivasan D, Grace M, Jiang XX. Comprehensive and efficient protection of kernel control data. IEEE Trans. on Information Forensics and Security, 2011,6(4):1404−1417. [doi: 10.1109/TIFS.2011.2159712]

[14] Baliga A, Ganapathy V, Iftode L. Detecting kernel-level rootkits using data structure invariants. IEEE Trans. on Dependable and Secure Computing, 2011,8(5):670−684. [doi: 10.1109/TDSC.2010.38]

[15] Vogl S, Gawlik R, Garmany B, Kittel T, Pfoh J, Eckert C, Holz T. Dynamic hooks: Hiding control flow changes within non-control data. In: Proc. of the 23rd USENIX Security Symp. Berkeley: Usenix, 2014. 813−828.

[16] Donghai T, Xuanya L, Changzhen H, Huaizhi Y. OPKH: A lightweight online approach to protecting kernel hooks in kernel modules. China Communications, 2013,10(11):15−23. [doi: 10.1109/CC.2013.6674206]

[17] Zhu F. Integrity-Based kernel malware detection [Ph.D. Thesis]. Florida International University, 2014.

[18] Rhee J, Xu D. LiveDM: Temporal mapping of dynamic kernel memory for dynamic kernel malware analysis and debugging. Technical Report 2010-02. Purdue University at West Lafayette, 2010. 1−20.

[19] Bergeron J, Debbabi M, Desharnais J, Erhioui MM, Lavoie Y, Tawbi N. Static detection of malicious code in executable programs. In: Proc. of the Symp. on Requirements Engineering for Information Security. 2001. 184−189.

[20] Yin H, Song D, Egele M, kruegel C, Kirda E. Panorama: Capturing system-wide information flow for malware detection and analysis. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2007. 116−127. [doi: 10.1145/1315245. 1315261]

[21] Xu L, Su Z. Dynamic detection of process-hiding kernel rootkit. Technical Report, CSE-2009-24, University of California at Davis, 2009. 1−12.

[22] Li B, Wo TY, Hu CM, Li JX, Wang Y, Huai JP. Hidden OS objects correlated detection technology based on VMM. Ruan Jian Xue Bao/Journal of Software, 2013,24(2):405−420 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4265.htm [doi: 10.3724/SP.J.1001.2013.04265]

[23] Petroni N, Fraser T, Walters A, Arbaugh WA. An architecture for specification-based detection of semantic integrity violations in kernel dynamic data. In: Proc. of the 15th Conf. on USENIX Security Symp. Berkeley: Usenix, 2006. 289−304.

[24] Ren P, Wang X, Wu C, Zhao B, Sun H. A semantic-based malware detection system design based on channels. information and communication technology. Cambridge: Berlin, Heidelberg: Springer-Verlag, 2014. 653−662. [doi: 10.1007/978-3-642-55032-4_67]

[25] Cozzie A, Stratton F, Xue H, King ST. Digging for data structures. In: Proc. of the 8th USENIX Conf. on Operating Systems Design and Implementation. Berkeley: Usenix, 2008. 255−266.

[26] Wang XL, Wang ZL, Sun YF, Liu Y, Zhang BB, Luo YW. Detecting memory leak via VMM. Chinese Journal of Computers, 2010,33(3):463−472 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2010.00463]

[27] Rhee J, Riley R, Lin Z, Jiang X, Xu D. Data-Centric OS kernel malware characterization. IEEE Trans. on Information Forensics and Security, 2014,9(1):72−87. [doi: 10.1109/TIFS.2013.2291964]

[28] Shinagawa T, Eiraku H, Tanimoto K, hasegawa S, Hirano M, Kourai K, Oyama Y, kawai E, Kono K, Chiba S, Shinjo Y, Kato K. Bitvisor: A thin hypervisor for enforcing I/O device security. In: Proc. of the 2009 ACM SIGPLAN/SIGOPS Int'l Conf. on Virtual Execution Environments. New York: ACM Press, 2009. 121−130. [doi: 10.1145/ 1508293.1508311]

附中文参考文献:

[6] 陈平.代码复用攻击与防御技术研究[博士学位论文].南京:南京大学,2012.

[22] 李博,沃天宇,胡春明,等.基于 VMM 的操作系统隐藏对象关联检测技术.软件学报,2013,24(2):405−420. http://www.jos.org.cn/1000-9825/4265.htm [doi: 10.3724/SP.J.1001.2013.04265]

[26] 汪小林,王振林,孙逸峰,等.利用虚拟化平台进行内存泄露探测.计算机学报,2010,33(3):463−472. [doi: 10.3724/SP.J.1016.2010.00463]

陈志锋(1986－),男,福建漳州人,博士生,主要研究领域为信息安全,可信计算.

张平(1969－),女,博士,副教授,CCF 专业会员,主要研究领域为并行识别,并行编译,信息安全.

李清宝(1967－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为信息安全,可信计算.

丁文博(1985－),男,讲师,CCF 专业会员,主要研究领域为信息安全.