基于系统调用的 Linux 进程行为画像

谢龙龙[1]，龚俭[1]

（1.东南大学网络空间安全学院，南京，210000）

摘　要：本文使用 audit 模块采集特权进程运行时产生的系统调用信息，并以此为信息源使用一阶齐次马尔科夫链模型对特权进程进行画像。对画像结果进行分析，得出特权进程运行时使用的系统调用较少且系统调用之间转移比较稳定的结论。为接下来使用系统调用信息对特权进程进行异常检测打下理论基础。

关键词：系统调用；马尔科夫链模型；进程行为画像

Linux process behavior profile based on system calls

Xie Longlong[1], Gong Jian[1]

(1. School of Cyber Science and Engineering, Southeast University, Nanjing, 210000)

Abstract: This paper uses the audit module to collect the system calls generated when the privileged process is running, and uses this as the information source to use the first-order homogeneous Markov chain model to profile the privileged process. Analyzing the results of the profile, it is concluded that fewer system calls are used when the privileged process is running, and the transfer between system calls is relatively stable. Lay a theoretical foundation for the subsequent use of system calls to perform anomaly detection on privileged processes.

Key words: System calls; Markov chain model; Process behavior profile

1 引言

根据我国国家互联网信息办公室发布的《2020 年上半年我国互联网网络安全监测数据分析报告》[1]显示国家信息安全漏洞共享平台（CNVD）收录通用型安全漏洞同比大幅增长 89.0%，其中应用程序漏洞占比最高，高达 48.5%。正常情况下，程序运行时产生的系统调用信息是稳定的，但是当进程遭到攻击或因漏洞被入侵后，其运行时产生的系统调用序列与正常时会有一定的差异[2]，因此研究人员可以根据进程运行时使用的系统调用信息对进程进行异常检测。

自从 Forrest 等人在 1996 年的文献[2]中首次提出使用系统调用序列对进程进行异常检测以来，基于系统调用序列的进程行为分析方法成为一种最普遍的基于主机的异常检测方法[3]。Warrender 等人在文献[4]中首先提出使用隐马尔可夫模型（HMM）进行入侵检测，并通过与 stide、t-stide 等方法对比，得出 HMM 检测效果较好但是更耗时的结论。Han 等人在文献[5]中提出基于进化神经网络异常检测方法，与传统神经网络方法相比，该方法能够在较短的时间内获得更优结构与权重，且实验结果表明该方法的准确性也要高于传统的方法。文献[6]提出了一种改进的基于二阶隐马尔可夫模型的检测方法，该方法重点利用系统调用局部规律来建模，在减少了训练时间的同时还增加了准确率。文献[7]介绍了一种多探测器 ADS，它可以在接收器工作特征（ROC）空间中使用布尔组合有效地组合来自异构探测器（例如 STIDE，HMM 和 OCSVM）的决策，以减少错误警报。

系统调用是操作系统提供给用户程序的一组特殊接口，用户程序可以通过这些接口使用操作系统内核提供的服务，这样在保护内核的同时还能让开发人员更好的使用内核层提供的功能[8]。进程在运行时会产生大量的系统调用信息，如果记录主机中所有进程运行时使用的系统调用信息会给主机带来极大的负载，消耗主机大量的计算资源与存储资源。攻击者往往通过利用设置 setuid 或 setgid 位的应用程序中的漏洞进行提权操作，从而使入侵者代码在

作者简介：谢龙龙，（1995-），男，硕士研究生，E-mail: llxie@njnet.edu.cn；龚俭，（1957-），男，教授，E-mail: jgong@njnet.edu.cn.

特权用户的上下文中运行[9]。因此，本文只记录特权进程运行时产生的系统调用信息，这样能够在大大减少记录信息量的同时提升检测准确性。

从上述文献中可以发现，研究人员往往只是使用系统调用检测进程是否异常，而并没有详细分析进程系统调用行为。本文旨在通过使用系统调用信息对特权进程进行画像并分析画像结果，为基于系统调用的进程异常检测提供理论支持。

2 Linux 特权进程系统调用信息采集

在 Linux 系统中，rsyslog 为默认的日志子系统，拥有强大的日志记录功能，能够记录主机上系统、应用有关的运行信息以及各种安全、认证事件信息。但是，rsyslog 属于应用层，而进程的系统调用属于进程内核层的信息，因此需要使用 audit 审计工具将进程系统调用信息写入审计日志中。

audit 是内核中的一个模块，在大部分 Linux 系统中都是默认安装的，主要记录系统调用和文件访问信息。audit 模块由应用层的一个应用程序 auditd 控制，audit 产生的数据也都会传送到 auditd 中。默认情况下 audit 是不设置任何规则的，也不审计任何事件，因此需要通过在 audit.rules 中设置规则以明确具体审计哪些事件。本文通过设置以下规则只记录特权进程的系统调用信息，详细规则内容如下所示：

-a exit,always -S all -F uid!=0 -F suid=0

-a exit,always -S all -F gid!=0 -F sgid=0

特权进程是那些设置 setuid 或 setgid 的应用程序运行时产生的进程，这些进程有一个特点，那就是运行该进程的用户属于普通用户即 uid!=0，但是该进程运行时的有效用户为超级用户即 suid=0。基于此，本文设置了以上的两条规则记录所有特权进程的系统调用信息。规则具体含义如表 1 所示：

表 1 audit 审计规则说明

规则	规则说明
-a exit,always	添加规则，总是在系统调用退出时记录它
-S all	所有类型系统调用
-F uid!=0 -F suid=0	只审计特权进程

基于以上规则，audit 就能以日志的形式记录特权进程运行时产生的系统调用信息，其中日志内容字段比较多，本文中只关注那些能够帮助解析出系统调用序列的字段，关注的字段与具体说明如表 2 所示。

表 2 审计日志字段说明

字段	说明
type=SYSCALL	表明本条日志记录进程系统调用信息
1609055532.857:23	当前系统调用执行的时间戳以及事件
68828	ID
syscall=11	表示当前系统调用函数为 munmap
pid=13457	进程 ID
ses=4294967295	会话 ID
exe="/bin/su"	应用程序可执行文件路径

其中时间戳以及事件 ID 用于确定当前系统调用执行的时间以及各个系统调用之间的前后顺序。pid 和 ses 能唯一确定特权进程的一次执行，确定一个系统调用序列。exe 表示是哪个应用执行产生的此进程。

本文首先使用 Logstash 实时采集特权进程运行时产生的审计日志，并使用过滤器将审计日志解析成 Json 格式，最后将解析后的日志统一输出到 Elasticsearch 集群中保存。随后编写脚本定期解析 ES 集群中的审计日志，将解析后得到的系统调用序列保存到 Mongodb 中以备后续使用，具体采集流程如图 1 所示。

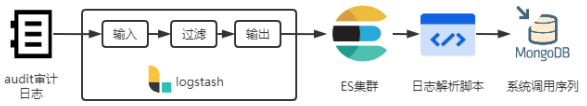

图 1 系统调用数据采集与解析流程

基于以上分析，根据获取的 audit 审计日志，能够解析出特权进程一次运行产生的系统调用序列，解析结果如下所示：

（open, read, mmap, getrlimit, …, close, close）

特权进程在不同的目录下不同的用户运行，以及使用时跟随不同的参数，产生的系统调用序列可能都有一定的差异。因此需要尽可能多的收集特权进程在正常环境下运行时产生的系统调用信息，这样才能保证画像结果的正确性与完整性。

3 基于系统调用的画像方法

3.1 系统调用分析

不同的 Linux 版本，Linux 系统中的系统调用个数不尽相同，但是一般在 300 到 400 之间。本文以实验室一台主机 A 中的一个特权进程为例分析特权进程的系统调用信息，该 Linux 主机中共有 333 不同的系统调用。

本文收集了主机 A 中的特权进程 su 执行了 65 次后的系统调用信息，65 次进程执行产生 65 个系统调用序列，共约 3 万次的系统调用。对特权进程 su 的系统调用进行分析后发现该进程只使用了 52 个不同的系统调用，且只有 17 个系统调用占比超过 1%。67%的系统调用占总调用次数的比例低于 1%。具体的所有使用的系统调用所占比例如图 2 所示。

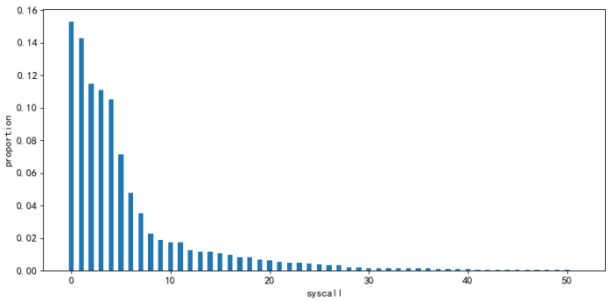

图 2 特权进程 su 系统调用概率分布图

3.2 使用马尔科夫链模型对进程行为画像

本文使用一阶齐次马尔科夫链模型对每个特权进程的系统调用行为进行画像，其画像过程就是根据特权进程历史上在正常环境中执行时产生的系统调用序列作为训练数据，计算出马尔科夫链模型的初始状态概率矢量π以及状态转移矩阵 A。

3.2.1 马尔科夫链模型状态的确定

基于上节对特权进程系统调用信息进行的分析，能够发现在正常情况下一个特权进程在运行时并没有使用所有的系统调用。相反地，只使用了几十种，且大部分系统调用占总系统调用次数的比例低于 1%，因此对一个特权进程使用马尔科夫链模型进行画像不能单纯的把每一种系统调用当做一种状态，因为这样会产生很多无意义的 0 值。具体的，本文确定马尔科夫链模型的状态分为以下几个步骤：

1) 获取该特权进程在正常运行时产生的系统调用序列信息作为训练数据。

设该特权进程的正常行为数据包含 M 个序列，记为 $T = (R_1, R_2, \dots, R_M)$，其中第 i 个序列为 $R_i = (s_1^i, s_2^i, \dots, s_{r(i)}^i)$，r(i)表示第 i 个系统调用序列共有 r(i)个系统调用。本文定义 r = r(1) + r(2) + … + r(M)，r 为正常训练数据中系统调用的总个数。

2) 将训练数据 T 中所有不同的系统调用提取出来，并计算它们的频率。

设正常训练数据中共有 N 个不同的系统调用，分别记为 $s^*_1, s^*_2, \dots, s^*_N$，并设第 j 个系统调用 s^*_j 在训练数据中出现的概率为 P_j，其计算公式为：

$$P_j = C_j / r \ , 1 <= j <= N \quad (1)$$

其中 C_j 为 s^*_j 在训练数据中出现的次数。

3) 设定一个频率阈值 u，将 $s^*_1, s^*_2, \dots, s^*_N$ 中出现频率大于等于阈值 u 的系统调用提取出来。假设共有 W 个（W <= N）系统调用的频率大于等于 u，分别记为 $s'_1, s'_2, \dots, s'_W$，它们的频率分别记为 $P'_1, P'_2, \dots, P'_w$，其中 P'_k 为 s'_k 在训练集中出现的频率。

4) 最终确定马尔科夫链的状态个数为 W+1，状态集合确定为 $O = \{s'_1, s'_2, \dots, s'_W, s'_{W+1}\}$。$s'_{W+1}$ 为附加状态，它对应于所有的在训练数据中出现频率小于阈值 u 的系统调用，其频率为 $1 - (P'_1 + P'_2 + , \dots , + P'_w)$。

3.2.2 马尔科夫链模型参数的计算

马尔科夫链的初始概率矢量$\pi = (\pi_1, \pi_2, \dots, \pi_{W+1})$用于描述特权进程正常运行时各个系统调用在初始时刻出现的概率，状态转移矩阵 $A = (a_{ij})_{(W+1)*(W+1)}$用于描述各个系统调用之间的时序相关性。其中初始概率矢量π的计算公式为：

$$\pi_i = \begin{cases} P'_i, & 1 \le i \le W \\ 1 - \sum_{j=1}^{W} P'_j, & i = W + 1 \end{cases} \quad (2)$$

将训练数据 T 中每个系统调用都与上文中求得的状态集合中的状态一一对应，若系统调用 s'_i 向 s'_j 转移的次数为 Z_{ij}，Z_{ij} 即为系统调用序列（s'_i, s'_j）在训练数据中的出现次数。设在训练数据中系统调用 s'_i 向各个系统调用转移的总次数为 Y_i，则 a_{ij} 的计算公式为：

$$a_{ij} = Z_{ij} / Y_i \quad (3)$$

基于以上对马尔科夫链模型状态的确定以及马尔科夫链模型参数的计算，本文就完成了特权进程系统调用行为的画像。

3.3 画像结果

本文使用一阶齐次马尔科夫链模型，利用特权进程运行时产生的系统调用信息对特权进程进行画像。其画像结果为马尔科夫链模型的参数信息，即初始状态概率矢量π以及状态转移矩阵 A。本文以上述提到的特权进程 su 为例对画像结果进行描述。其中π为上文中确定的各个状态的概率，A 为状态转移矩阵，结果分别如图 3、图 4 所示。

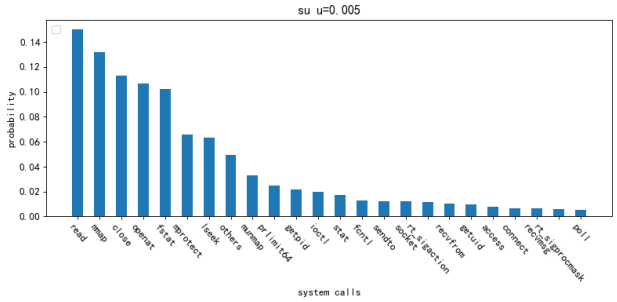

图 3 初始状态概率矢量 π

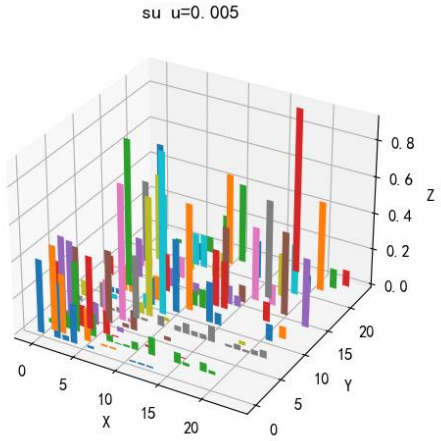

图 4 状态转移矩阵 A

其中图 4 中 X 轴、Y 轴中的 0 代表图 3 中的 read 状态，1 代表图 3 中的 mmap 状态，Z 轴代表状态 A 到状态 B 的转移概率。如图 4 中的（0, 2）点对应 Z 轴的值为 0.36 代表状态 read 转到状态 close 的概率为 0.36。

4 进程行为画像分析

本文分别对网络中心主机 A 中的 su、ping 进程进行画像。su 和 ping 都为 Linux 中常用的命令，su 用于切换用户，ping 用于测试网络连接是否畅通，它们运行时产生的进程都属于特权进程。对于 su 和 ping 命令来说，不同的用户使用，使用时跟随不同的参数，运行是否成功以及运行的时长产生的系统调用序列长度、使用的系统调用以及系统调用之间的关系都不尽相同。因此，本文分别取 su 进程运行 40 与 70 次产生的系统调用序列对 su 进程进行画像，取 ping 进程运行 40 次产生的系统调用序列对 ping 进程进行画像，分别用 su-40、su-70、ping-40 代表它们的画像结果。

su 进程运行 40 次产生 40 个系统调用序列，共 25385 次系统调用，使用了 52 种不同的系统调用。su 进程运行 70 次产生 70 个系统调用序列，共 49345 次系统调用，同样也使用了 52 种不同的系统调用。ping 进行运行 40 次产生 40 个系统调用序列，共 10187 次系统调用，使用了 39 种不同的系统调用。下表中分别列出了以上 3 个画像结果中排名前十的状态，以及各个状态的概率，具体结果如表 3 所示。

表 3 画像结果中排名前 10 状态

su-40		su-70		ping-40	
状态	概率	状态	概率	状态	概率
read	0.148	read	0.147	mmap	0.145
mmap	0.130	mmap	0.129	openat	0.092
close	0.112	close	0.113	close	0.080
openat	0.106	openat	0.107	mprotect	0.073
fstat	0.102	fstat	0.102	recvmsg	0.073
lseek	0.067	lseek	0.070	fstat	0.070
mprotect	0.065	mprotect	0.064	read	0.057
munmap	0.032	munmap	0.030	sendto	0.048
prlimit64	0.027	prlimit64	0.027	write	0.045
getpid	0.023	getpid	0.023	socket	0.030

从表 3 中可以明显看出，su-40 与 su-70 前 10 个状态一模一样，且概率也几乎一样。而 ping 进程与 su 进程的状态相差就比较大，除了状态相差比较大之外，相同状态的概率相差也比较大。ping 进程排名靠前的有很多网络操作相关的系统调用，如 recvmsg、sendto、socket，而这些系统调用在 su 进程中出现概率非常低。

从 3.2.1 节中能够得知阈值 u 大小的选择决定了马尔科夫模型中状态的个数，本文通过调节 u 的大小观察上述 3 个画像状态个数的变化。

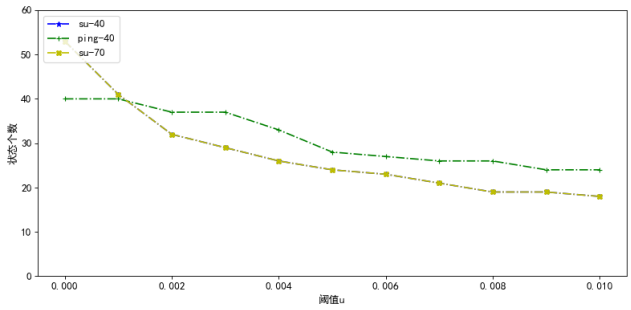

图 5 状态个数与 u 大小关系

从图 5 中可以发现随着阈值 u 的增大，马尔科夫链模型中的状态个数一直在减少，当 u 为 0.005 时开始趋于稳定，且 su 进程两条折线完全重合。这说明特权进程一般只较多的使用少量的系统调用，大部分使用的系统调用比例比较低。

根据图 4 可以发现并非所有状态都会向所有状态转移，大部分状态都只会向某几个固定的状态转移。如 su-40，当阈值 u 取 0.005 时共有 24 个状态，其中只有两个状态向超过 10 个状态转移，且有一个为 s'_{W+1} 状态即“others”状态；超过 54%的状态只向不超过 5 个状态转移，而 ping-40 这一比例超过 78%。

将 su 进程一个新的系统调用序列放入 su-40 与 su-70 画像模型中，将 ping 进程一个新的系统调用序列放入 su-40 画像模型中计算。计算方法为使用固定大小为 6 步长为 1 的滑动窗口[2]，依次计算系统调用序列中各个滑动窗口内的状态转移概率，每个窗口的状态转移概率为窗口内各个状态转移概率相乘。结果如图 6 所示。

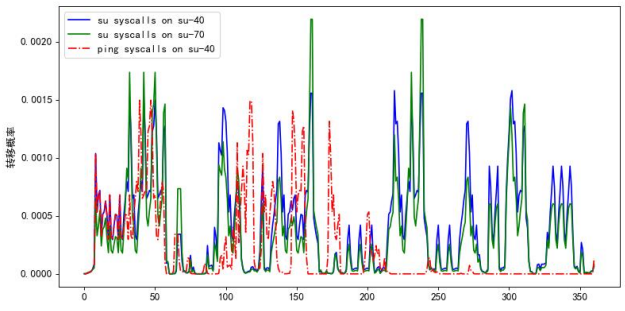

图 5 系统调用序列在画像模型上状态转移概率

红色点划线为 ping 进程产生的系统调用序列在 su-40 画像模型中计算的结果，可以看到前半部分大部分窗口内的状态转移概率值都比较高，与 su 进程产生的系统调用序列相差不大；但是到了后半部分概率值却连续的大范围低于 su 进程产生的系统调用序列。而 su 进程的系统调用序列在 su-40 与 su-70 上的状态转移概率曲线相似度非常高，很多地方都高度重合。

基于以上分析，我们得出以下三点结论：

（1）特权进程运行时使用的系统调用种类比较少，且各个系统调用之间的转移比较稳定。

（2）不同特权进程运行时使用的系统调用种类差距比较大，且得到的画像结果相差也比较大。

（3）使用少量的系统调用序列就能得到一个特权进程比较完整的画像结果。

5 总结

本文首先使用 audit 模块收集特权进程运行时产生的系统调用信息，经过预处理后将系统调用信息解析成一个个系统调用序列。然后使用一阶齐次马尔科夫链模型对特权进程运行时系统调用行为进行画像，画像结果为马尔科夫链模型的参数信息，即初始状态概率矢量 π 以及状态转移矩阵 A。最后对特权进程画像结果进行分析得出特权进程运行时使用的系统调用较少且状态转移比较稳定的结论。为接下来使用特权进程运行时产生的系统调用信息对进程进行异常检测打下理论基础。

参考文献

[1] 2020 年上半年我国互联网网络安全监测数据分析报告[EB/OL].http://www.cac.gov.cn/2020-09/26/c_1602682854845452.htm, 2020.

[2] Forrest S, Hofmeyr S A, Somayaji A, et al. A sense of self for unix processes[C]//Proceedings 1996 IEEE Symposium on Security and Privacy. IEEE, 1996: 120-128.

[3] Liu M, Xue Z, Xu X, et al. Host-based intrusion detection system with system calls: Review and future trends[J]. ACM Computing Surveys (CSUR), 2018, 51(5): 1-36.

[4] Warrender C, Forrest S, Pearlmutter B. Detecting intrusions using system calls: Alternative data models[C]//Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344). IEEE, 1999: 133-145.

[5] Han S J, Cho S B. Evolutionary neural networks for anomaly detection based on the behavior of a program[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2005, 36(3): 559-570.

[6] 吴鑫, 严岳松, 刘晓然. 基于改进 HMM 的程序行为异常检测方法[J]. 信息网络安全, 2016(09):108-112.

[7] Khreich W, Murtaza S S, Hamou-Lhadj A, et al. Combining heterogeneous anomaly detectors for improved software security[J]. Journal of Systems and Software, 2018, 137: 415-429.

[8] 鸟哥. 鸟哥的 Linux 私房菜,基础学习篇[M]. 人民邮电出版社, 2010.

[9] 龚俭, 杨望. 计算机网络安全导论-第 3 版[M]. 东南大学出版社, 2019.