北京大學
PEKING UNIVERSITY

Auditing Frameworks Need Resource Isolation: A Systematic Study on the Super Producer Threat to System Auditing and Its Mitigation

Peng Jiang, Ruizhe Huang, Ding Li*, Yao Guo, Xiangqun Chen

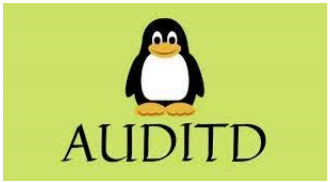
MOE Key Lab of HCST, School of Computer Science, Peking University

Jianhai Luan, Yuxin Ren and Xinwei Hu

Huawei Technologies

Provenance Data & Auditing Framework

- **Auditing framework** is the fundamental component of a provenance analysis system, which collects provenance data (e.g. syscalls) from OS kernels. It can be taken as “provenance collector”.
- SoTA: Sysdig, LTTng, Linux Audit, Camflow.



CamFlow

Practical Linux Provenance

7 followers Canada

Super Producer Threat

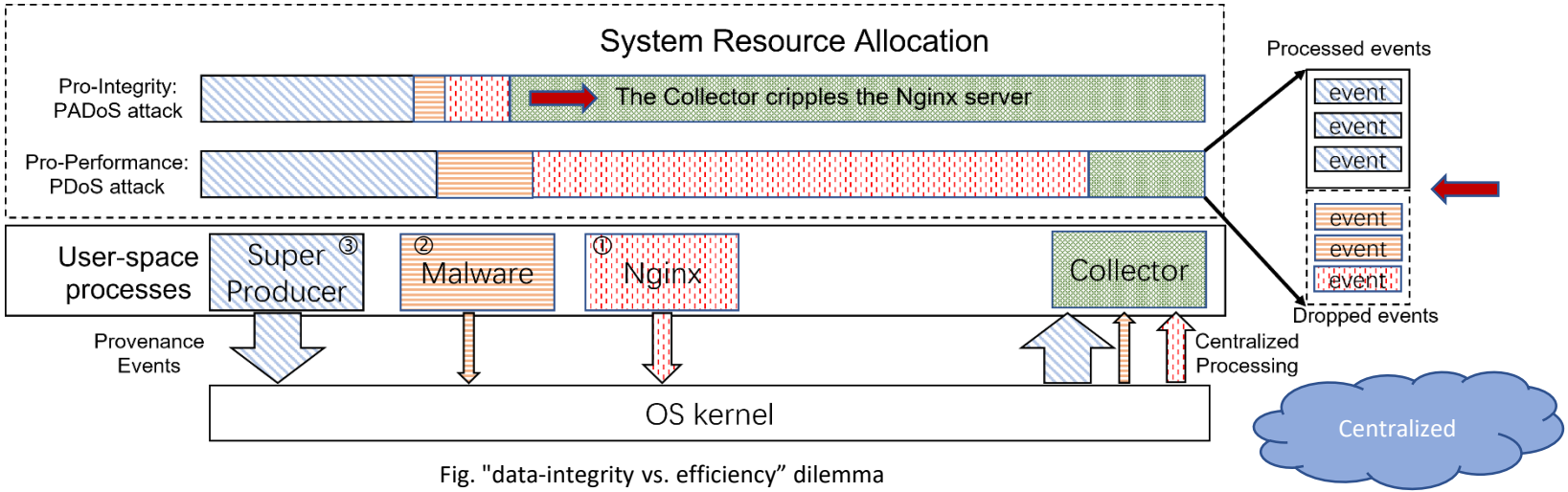


Fig. "data-integrity vs. efficiency" dilemma

- Three types of processes: Super producer and Malware and Nginx
- PDos Attack causes event dropping (*Sysdig, Lttng and Linux Audit*)
- PADos Attack causes performance degrade (*Camflow*)

Challenges Under This Architcture

- Straightforword solution based on nowadays architecture
  - Suppress the super producer's generation speed with some specified threshold. For example Cgroups.
- Challenges
  - Hard to determine an effective threshold.
  - Attackers can use a set of super producers to avoid reaching the threshold.
- This solution is impractical !

Mitigation – Insights

- The centralized architecture of collectors **breaks the logic and resource isolation** between processes.
- Audit logs should be the process's own data and be processed using its own resource quota.

ISOLATION !

Design of NODROP

Insight: We need logic and resource isolation.

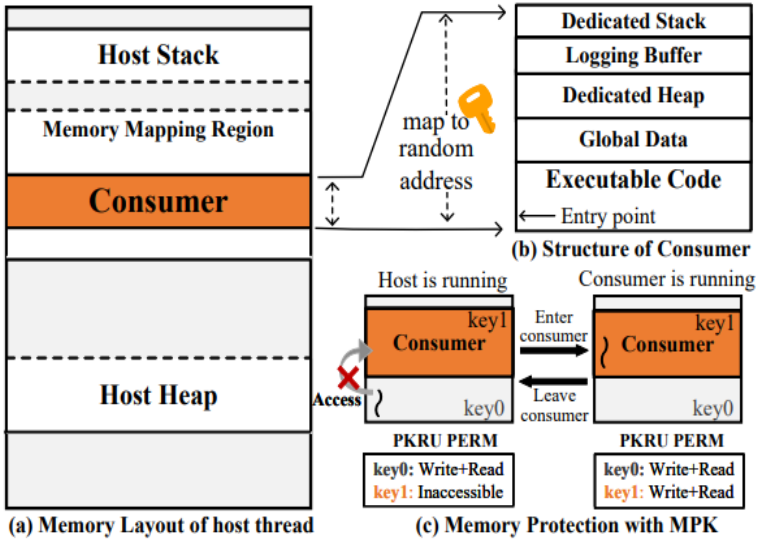
Design Goal: *Zero Data Lost, Performance Isolation, Low Overhead.*

Isolation Strategy:

- *Self-consuming execution.* Each running thread digests the provenance data it generates. *(logic isolation)*
- *Synchronized logging buffer.* Allocate a dedicated logging buffer for each thread and synchronously digests provenance data within it. *(data isolation)*

Design of NODROP

- We proposed a threadlet-based design based on these two principles, called NODROP.
  - Instantiates the provenance data processing logic (consumer) as a threadlet.
  - Consumer is insterted into the memory of the current running thread.
  - Consumes the provenance data stored in the dedicated logging buffer.



Memory protection:
Address randomness and MPK

High-Level Workflow

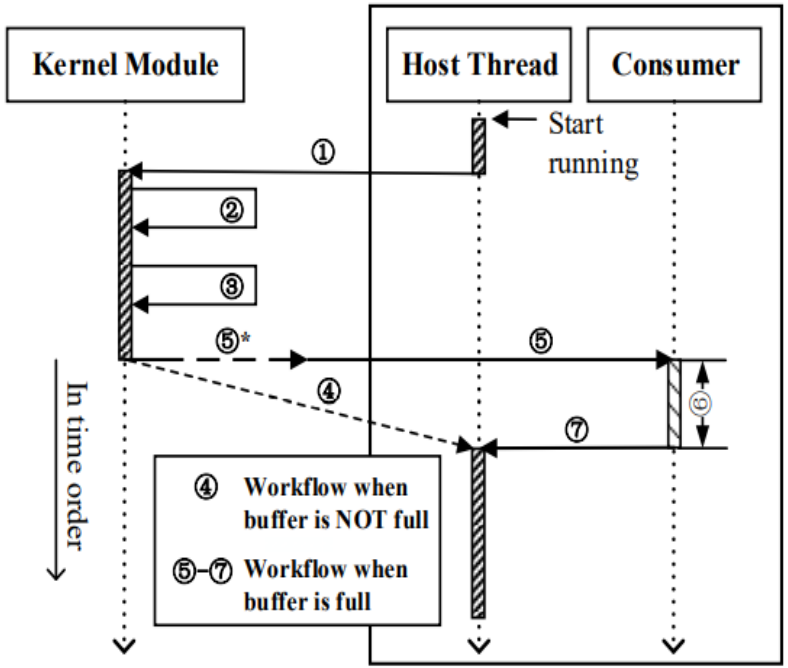
S1: The kernel captures a system call (①) and executes it (②). System call is recored in a dedicated in-kernel logging buffer (③).

S2: Buffer is not full: returns control to the host thread (④). Buffer is full or the thread exits: control is passed to the consumer (⑤).

S3: Consumer is instrumented into the running thread(⑤∗).

S4: Consumer process the transferred provenance data (⑥).

S5: Control is returned to the original thread (⑦).

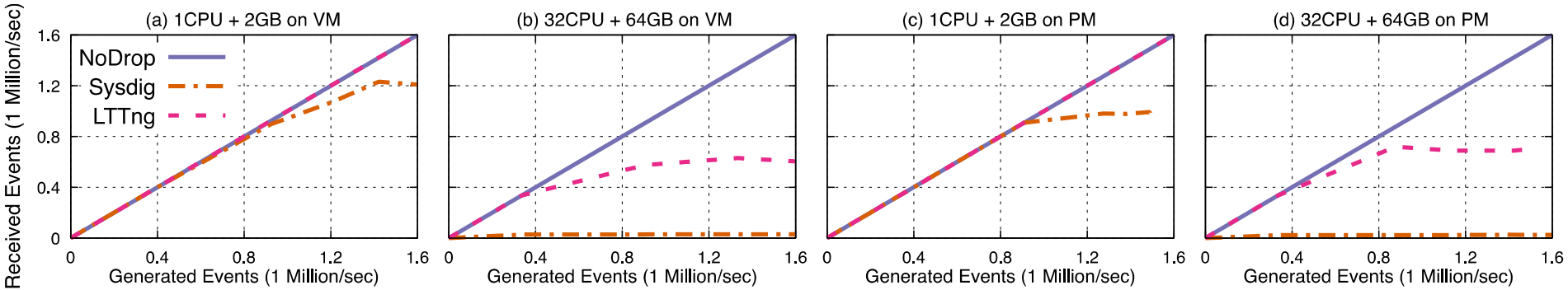


Evaluation

Research Questions

- RQ1: Can NODROP avoid dropping provenance data?
- RQ2: Can NODROP prevent a super producer from slowing down other applications?
- RQ3: What is the runtime overhead of NODROP?
- RQ 4: Can data reduction techniques address the super producer threat?
- RQ 5: Can increasing the buffer size address the super producer threat?

RQ1: Preventing the PDos attacks

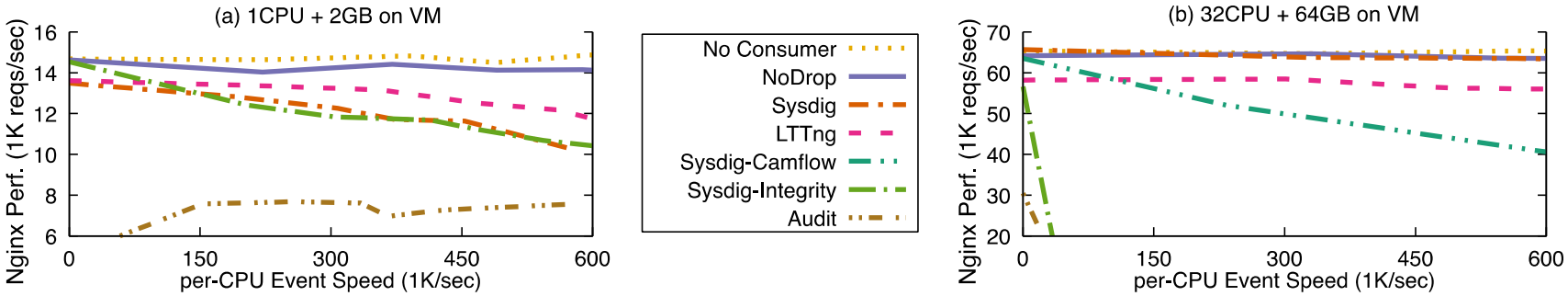


- NODROP prevents the PDoS attack.
- NODROP has no events dropping.
- The successful attack rate for NODROP is ZERO.

Table. Attack success rate of the PDoS attack (#success / #attempts)

	Sysdig	LTTng	Linux Audit	NODROP
Default	120/120	107/120	120/120	**0/120**
Cgroup	115/120	107/120	120/120	**0/120**

RQ2: Preventing the PADos attacks



- When the workload of the super producer grows, NODROP maintains the performance of applications stable regardless of the hardware configurations.
- NODROP is robust against PADoS attacks.

RQ3: Runtime Overhead

Application	Collector	C1	C4	C5	C8
Nginx	NODROP	9.80	3.60	11.35	4.84
	Sysdig	55.30	5.50	37.64	7.20
	DIFF	**-29.30**	**-1.80**	**-19.10**	**-2.20**
Redis	NODROP	8.90	3.10	8.66	2.42
	Sysdig	21.00	5.00	21.00	5.70
	DIFF	**-11.10**	**-2.00**	**-10.20**	**-3.10**
Postmark	NODROP	25.50	19.10	30.60	20.65
	Sysdig	96.30	8.60	95.80	13.50
	DIFF	**-36.00**	**9.70**	**-33.30**	**6.30**
Django (Python)	NODROP	1.30	2.00	1.30	-0.20
	Sysdig	1.10	2.30	1.10	0.30
	DIFF	0.30	-0.30	0.20	-0.50
http (Golang)	NODROP	14.10	2.20	14.66	2.71
	Sysdig	78.90	2.20	65.70	2.20
	DIFF	**-36.20**	0.10	**-30.80**	0.50
OpenSSL	NODROP	0.60	0.10	0.20	0.70
	Sysdig	0.60	0.10	0.20	0.60
	DIFF	0.10	0.00	0.00	0.10
7-ZIP	NODROP	0.30	0.80	1.20	0.70
	Sysdig	0.20	0.70	1.20	0.70
	DIFF	0.10	0.10	0.00	0.00
PostgreSQL	NODROP	7.05	3.80	10.20	4.70
	Sysdig	15.20	4.70	17.40	4.90
	DIFF	**-7.61**	-0.80	**-6.50**	-0.19

Table 3: Performance scores of *lmbench*. *All values are shown as percentages relative to Sysdig. The negative value means* NODROP *is faster than Sysdig.*

Configurations	C1	C4	C5	C8	Ave
Syscall Tests					
NULL syscall	-8.1%	-17%	-8.3%	-7.9%	-10.3%
stat	-9.0%	+5.5%	-1.8%	-0.6%	-1.5%
fstat	+4.2%	-1.7%	+1.7%	+1.6%	+2.3%
open/close file	-6.1%	-2.9%	-0.3%	-1.8%	-2.8%
read file	+7.4%	+7.1%	+4.5%	+7.2%	+6.6%
write file	+7.7%	+7.2%	+12.5%	+12.1%	+9.9%
File Access					
file create (0K)	-15.8%	-7.1%	-10.0%	+2.7%	-7.5%
file delete (0K)	+0.5%	+3.0%	-0.7%	-0.9%	+0.5%
file create (10K)	+0.1%	+2.9%	-3.7%	-0.8%	-0.4%
file delete (10K)	+4.7%	+1.5%	-0.9%	-0.4%	+1.2%
pipe	+3.0%	+0.8%	+6.9%	+1.3%	+3.0%
AF_UNIX	+3.8%	-10.5%	+5.3%	+10.1%	+2.2%

RQ4 and RQ5: Alternatives

- Log reduction techniques (CPR)
  - The kernel CPR can handle 2,000 events per second per core.
  - The super producer can easily generate 100,000 events per second.
  - The kernel CPR greatly blocks the running applications, amplifying the PADos attack.
- Increasing buffer size
  - Maximum size: 768M for Sysdig and LTTng and 77,000 messages for Linux Audit.
  - Still dropping when the generation speed per core reaches 1.6 million per second.

Configuration	C1	C4	C5	C8
Linux Audit	99.2%	99.6%	99.1%	99.4%
Sysdig	19.3%	86.1%	25.5%	88.1%
LTTng	0%	49.5%	0%	52.1%

Conclusion

- Existing auditing frameworks suffer from the super producer threat.
- We find that attackers can either disable existing provenance collectors or paralyze the whole system with a super producer.
- We propose a novel auditing framework, **NODROP**, that addresses the super producer threat by providing resource isolation.
- Our evaluation shows that NODROP introduces 6.30% lower application overhead on average across eight different hardware configurations than SOTA(Sysdig).

Related Pointers:

- Open-Source Repository:

  https://github.com/PKU-ASAL/NODROP

- Contact me:

  pengjiang_pku2020@stu.pku.edu.cn

Thank you and question?