Towards Provenance for Cybersecurity in Cloud-Native Production Infrastructure

FOSAD 2025 PhD Forum

Paul R. B. Houssel[1,2] Sylvie Laniepce[1] Olivier Levillain[2]

[1]Orange Research, Caen, France
[2]Institut Polytechnique de Paris, SAMOVAR, Télécom SudParis, Palaiseau, France

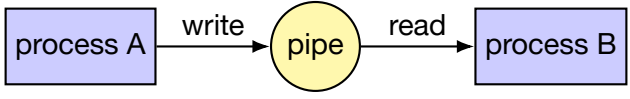
June 25, 2025

Provenance Graphs

- interactions between system subjects and objects
- understand system behavior, establish causality

Provenance Graphs

- interactions between system subjects and objects
- understand system behavior, establish causality
- threat detection
  - sub-graph embedding [1]
  - graph queries [2]
  - benign behavior model [1, 3]
- forensics
  - post-mortem root cause analysis [4]
  - active threat hunting [5]

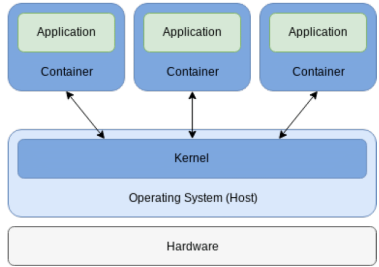


PhD Research Plan

Cloud-Native: An emerging infrastructure shift

Telemetry Collection for Cloud-Native production environments

- fine-grained, per-container telemetry
- distinguish container and host activity
- handle large system activity
- uniquely identify system objects



Telemetry Collection for Provenance

Location	Approach	Efficiency	Visibility	Safety	Portability	Example
user land	ptrace, fs snapshot	❍	❍	●	●	strace [6], ARTISAN [7]
kernel	integrated tool	●	●	●	❍	ftrace [8], auditd [9]
	kernel module	●	●	❍	●	CamFlow [4]
	eBPF	●	●	●	●	falco [10], tetragon [11]

Telemetry Collection for Provenance

Location	Approach	Efficiency	Visibility	Safety	Portability	Example
user land	ptrace, fs snapshot	❍	❍	●	●	strace [6], ARTISAN [7]
kernel	integrated tool	●	●	●	❍	ftrace [8], auditd [9]
	kernel module	●	●	❍	●	CamFlow [4]
	eBPF	●	●	●	●	falco [10], tetragon [11]

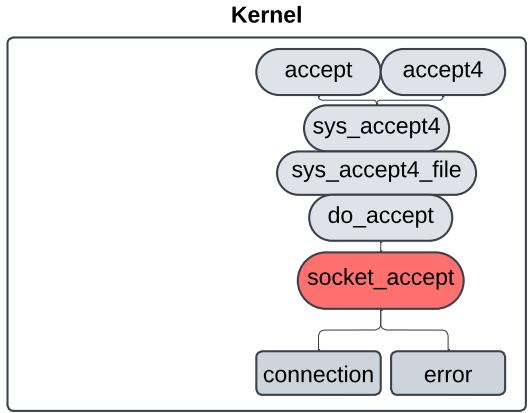
Telemetry Collection for Provenance

Location	Approach	Efficiency	Visibility	Safety	Portability	Example
user land	ptrace, fs snapshot	❍	❍	●	●	strace [6], ARTISAN [7]
kernel	integrated tool	●	●	●	❍	ftrace [8], auditd [9]
	kernel module	●	●	❍	●	CamFlow [4]
	eBPF	●	●	●	●	falco [10], tetragon [11]

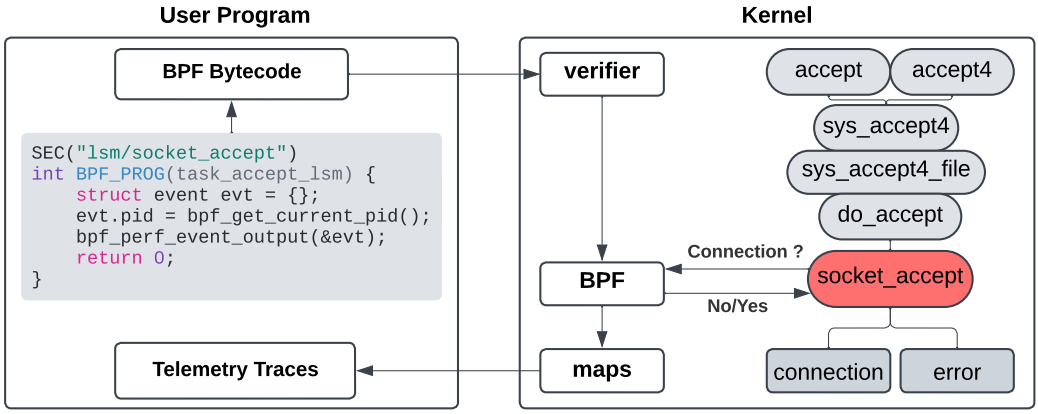
Telemetry Collection for Provenance

Location	Approach	Efficiency	Visibility	Safety	Portability	Example
user land	ptrace, fs snapshot	❍	❍	●	●	strace [6], ARTISAN [7]
kernel	integrated tools	●	●	●	❍	ftrace [8], auditd [9]
	kernel module	●	●	❍	●	CamFlow [4]
	eBPF	●	●	●	●	falco [10], tetragon [11]

Linux Security Module (LSM) hooks



Linux Security Module (LSM) hooks



Preliminary Results and Future Work

Coverage: sufficient system visibility for sound provenance ?

Coverage: sufficient system visibility for sound provenance ?

Objective

relationship between system calls and LSM hooks

Method

static analysis of Linux kernel source code (v6.13)

- cflow (call graphs) [12]
- cscope (code navigation) [13]

Coverage: sufficient system visibility for sound provenance ?

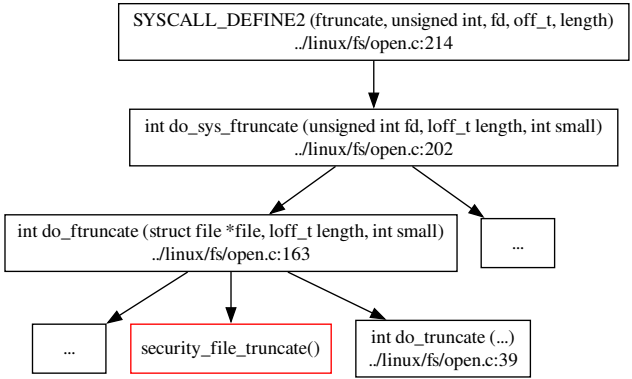
Objective

relationship between system calls and LSM hooks

Method

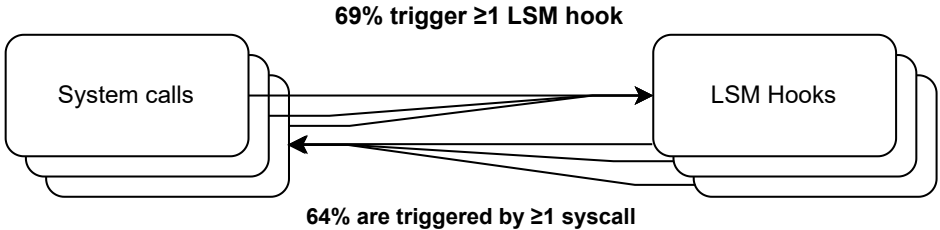
static analysis of Linux kernel source code (v6.13)

- cflow (call graphs) [12]
- cscope (code navigation) [13]



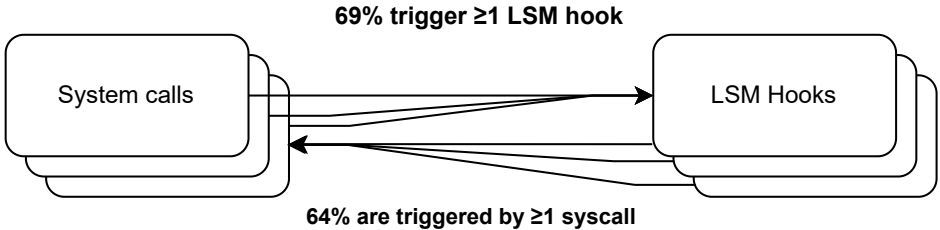
Coverage Analysis ($Syscall \leftrightarrow LSM$)

Limitations



Coverage Analysis ($Syscall \leftrightarrow LSM$)

Limitations



- indirect function calls
- function exported from a kernel module
- conditional branching is not considered

Stability across kernel versions compared to system calls

Stability across kernel versions compared to system calls

Objective

compare the rate of change in LSM hook interfaces vs. system call interfaces

Method

track Application Binary Interface (ABI) changes across kernel versions

- added functions
- removed functions
- signature changes

Stability across kernel versions compared to system calls

- backwards compatibility
- LSM and interface has grown substantially
- frequent changes:
  - additions
  - removals
  - renaming
  - argument modifications

Linux version	Release date	LSM hooks	Argument changes	System calls	Argument changes
2.6.12	2005-06-17	=131	–	=251	–
2.6.30	2009-06-09	+72/-22	19	+79/-0	8
3.1	2011-10-24	+17/-14	8	+17/-1	15
3.12	2013-11-03	+12/-4	12	+13/-0	16
4.1	2015-06-21	+7/-2	3	+9/-0	4
4.12	2017-07-02	+8/-3	19	+10/-0	5
5.1	2019-05-05	+23/-4	24	+37/-0	27
5.12	2021-04-25	+16/-2	26	+15/-1	13
6.1	2022-12-11	+10/-2	10	+8/-1	0
6.12	2024-11-17	+33/-8	20	+14/-1	7
6.13	2025-01-19	+6/-4	1	+4/-0	0

Performance overhead among eBPF programs at LSM operation equivalence (excluding tracepoint programs)

Activity context	Prog. type	Perf. overhead	Execution counts	Written traces
network	LSM kprobe tracing	1.07% 1.81% **0.90%**	50,002	50,002
file	LSM kprobe tracing	5.83% 6.39% **5.14%**	50,507	50,507
process	LSM kprobe tracing	0.74% 0.79% **0.47%**	50,035	50,035

Conclusion

Future Work

- refine coverage analysis
  - advanced static tools like Kayrebt [14]
  - fuzzing-based dynamic analysis [15]
- identify missing LSM hooks to cover kernel object
  - allocation
  - activity
  - information flow
- qualify the ABI changes

Towards Provenance for Cybersecurity in Cloud-Native Production Infrastructure

FOSAD 2025 PhD Forum

Paul R. B. Houssel[1,2] Sylvie Laniepce[1] Olivier Levillain[2]

[1]Orange Research, Caen, France
[2]Institut Polytechnique de Paris, SAMOVAR, Télécom SudParis, Palaiseau, France

June 25, 2025

References I

- T. Song, M. Organokov, L. Gulikers, G. Grassi, G. Carofiglio, and M. Meo, “Advancing Cloud-Native Cyber Threat Detection with Graph-Based Feature Engineering,” pp. 4291–4297, IEEE Computer Society, May 2025. ISSN: 2375-026X.
- W. Blair, F. Araujo, T. Taylor, and J. Jang, “Automated Synthesis of Effect Graph Policies for Microservice-Aware Stateful System Call Specialization,” in 2024 IEEE Symposium on Security and Privacy (SP), pp. 4554–4572, May 2024. ISSN: 2375-1207.
- B. Jiang, T. Bilot, N. E. Madhoun, K. A. Agha, A. Zouaoui, S. Iqbal, X. Han, and T. Pasquier, “ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems,” 2025.

References II

- T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, “Practical whole-system provenance capture,” in Proceedings of the 2017 Symposium on Cloud Computing, (Santa Clara, California, US), pp. 405–418, ACM, Sept. 2017.
- J. Li, R. Zhang, J. Liu, and G. Liu, “LogKernel: A Threat Hunting Approach Based on Behaviour Provenance Graph and Graph Kernel Clustering,” Security and Communication Networks, vol. 2022, no. 1, p. 4577141, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2022/4577141.
- “strace - the linux syscall tracer,” 1991.

References III

- L. Yu, Y. Ye, Z. Zhang, and X. Zhang, “Cost-effective Attack Forensics by Recording and Correlating File System Changes,” in Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), (Philadelphia, PA, USA), 2024.
- “ftrace - Function Tracer — The Linux Kernel documentation.”
- “auditd - The Linux Audit daemon,” 1994.
- “Falco: open source security tool for containers, kubernetes and cloud,” 2014.
  original-date: 2021-02-08T14:46:41Z.
- “Tetragon - eBPF-based Security Observability and Runtime Enforcement,” 2022.
- “Cflow - GNU Project - Free Software Foundation.”

References IV

- “cscope - interactively examine a C program,” 2002.
- L. Georget, F. Tronel, and V. V. T. Tong, “Kayrebt: An activity diagram extraction and visualization toolset designed for the Linux codebase,” in 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT), (Bremen, Germany), pp. 170–174, IEEE, Sept. 2015.
- D. Jones, “Trinity: A Linux system call fuzzer.”
- R. Guo and J. Zeng, “Phantom Attack: Evading System Call Monitoring,” 2021.

Stability Analysis (ABI Evolution)

Kernel	Release Date	# LSM Hook name changes	Modified function names	# LSM Hook Parameter changes	Modified Parameters	Total Syscalls	Changes from Previous	# Syscall Parameter changes
6.13	2025-01-19	+6/-4	current_getlsmprop_subj, lsmprop_to_secctx, inode_getlsmprop, cred_getlsmprop, task_getlsmprop_obj, ipc_getlsmprop ~~task_getsecid_obj~~, ~~ipc_getsecid~~, ~~current_getsecid_subj~~, ~~inode_getsecid~~	1	audit_rule_match: structlsm_prop*prop, ~~u32secid~~	+4/-0	removexattrat, getxattrat, listxattrat, setxattrat	0

eBPF attachements

Program type	Attach type	TOCTOU resistant [16]	Granularity					Stable	Kernel ≥
			system Calls	kfunc.	ufunc.	AC op.	cgroup		
LSM	LSM_MAC	●	❍	❍	❍	●	●	●	5.7
tracepoint	tracepoint	◗	●	◗	❍	❍	❍	●	4.7
kprobe	kprobe	◗	●	●	●	●	❍	❍	4.1
tracing	fentry	◗	●	●	●	●	❍	❍	5.1