CrackArmor: Nine AppArmor Flaws Enable Container Escape

Linux Kernel Vulnerabilities Break Container Isolation, Enable Privilege Escalation to Root

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-26

Key Takeaways

On March 12–13, 2026, Qualys Threat Research Unit (TRU) publicly disclosed nine security vulnerabilities collectively named CrackArmor, affecting Linux AppArmor — the mandatory access control (MAC) module that underpins container isolation on Ubuntu, Debian, SUSE, and derivatives [1]. The vulnerabilities have existed since Linux kernel v4.11 (2017) and span a range of severity classes: confused deputy attacks, use-after-free memory corruption, double-free conditions, kernel memory leaks, and uncontrolled recursion leading to kernel panic [2].

Two CVEs have been assigned — CVE-2026-23268 and CVE-2026-23269 — with seven additional identifiers pending assignment. Qualys researchers demonstrated full root escalation on Ubuntu 24.04.3 LTS via a use-after-free chain and on Debian 13.1 via a double-free exploit, both achieving kernel code execution [2]. Because AppArmor is the MAC enforcement layer for Docker's default security model and a widely deployed hardening mechanism for containerd-based Kubernetes runtimes, these vulnerabilities directly threaten container isolation on a significant portion of enterprise Linux deployments — one estimate, drawn from Qualys CyberSecurity Asset Management telemetry, places the affected population at 12.6 million enterprise instances [1][3].

Upstream kernel patches landed simultaneously with disclosure; distribution-level updates are available or pending for all supported Ubuntu and Debian releases. Organizations running containerized workloads should treat this as a priority patching event. Qualys withheld proof-of-concept exploit code to allow time for patch rollout, but the level of technical detail published is likely sufficient for skilled adversaries to develop independent exploits [1][2].

Background

What Is AppArmor?

AppArmor is a Linux Security Module (LSM) integrated into the kernel that enforces mandatory access control through per-process security profiles. Unlike discretionary access controls (which rely on file ownership and permissions), AppArmor confines applications by defining what files, capabilities, and

system calls each process is permitted to use, regardless of the process owner's identity. It is enabled by default on Ubuntu and SUSE distributions, and is used as an optional hardening layer on Debian and other derivatives [4].

In container environments, AppArmor plays a critical structural role. Docker applies a default AppArmor profile (`docker-default`) to every container, restricting access to sensitive kernel features and syscalls that would otherwise enable container breakout. Kubernetes inherits this behavior through containerd and other container runtimes. When AppArmor is functioning correctly, it provides a meaningful additional barrier between a compromised container process and the underlying host kernel. When it can be manipulated or bypassed, the isolation boundary is significantly weakened [5].

The CrackArmor Disclosure

Saeed Abbasi, Senior Manager at Qualys TRU, identified nine distinct vulnerabilities in AppArmor's kernel implementation and coordinated disclosure with Linus Torvalds' upstream kernel tree and major Linux distribution maintainers [1][2]. The vulnerabilities are rooted in AppArmor's profile management subsystem — specifically the code paths responsible for loading, replacing, and verifying policy profiles — which appears to have received less continuous security scrutiny than more commonly targeted kernel components, as the nine-year exposure window suggests [2]. The flaws were present in all distributions with AppArmor enabled running Linux kernel v4.11 or later, placing the exposure window at approximately nine years.

Security Analysis

The Nine Vulnerabilities

The CrackArmor vulnerability set covers five distinct vulnerability classes. Understanding each class is important because the attack chains that Qualys demonstrated combine several of them in sequence — the most severe paths require chaining multiple flaws to achieve root escalation.

Confused Deputy (CVE-2026-23268). The pseudo-filesystem entries at `/sys/kernel/security/apparmor/.load`, `.replace`, and `.remove` are exposed world-writable (mode 0666). This allows an unprivileged user to redirect the output of a privileged process — such as `sudo` or `Postfix` — into these files, effectively loading, replacing, or removing arbitrary AppArmor profiles without elevated permissions. An attacker exploiting this flaw can disable the AppArmor profile protecting a critical service, load a permissive profile for a target process, or remove

protections on a container process, collapsing its isolation boundary. Critically, this vulnerability also defeats Ubuntu's unprivileged user-namespace restrictions, which have served as a layered defense against local privilege escalation since their introduction [2].

Kernel Memory Disclosure (CVE-2026-23269). An unsafe macro in AppArmor's DFA (Deterministic Finite Automaton) pattern-matching engine increments a string pointer past null terminators during profile parsing. An attacker who can load crafted profiles can trigger this to read up to 64 KiB of kernel memory beyond the allocated buffer. The disclosed data includes KASLR-randomized kernel pointers, defeating kernel address-space layout randomization and enabling subsequent exploit stages to precisely target kernel data structures [2].

Use-After-Free (Local Privilege Escalation to Root). A race condition exists in the reference-counting logic for `aa_loaddata` structures allocated in the kmalloc-192 slab cache. When a file operation races against profile removal, the kernel can access freed memory. Qualys constructed a full exploitation chain that achieves root on Ubuntu 24.04.3 using this flaw in combination with the KASLR defeat from CVE-2026-23269 and page-table manipulation techniques — notably, this succeeds even with `CONFIG_RANDOM_KMALLOC_CACHES` enabled, a modern kernel hardening feature intended to complicate slab-cache-based exploitation [2].

Double-Free (Local Privilege Escalation to Root). In the `aa_replace_profiles()` function, namespace names (`ns_name`) are freed twice when profiles implicitly specify a namespace. This affects kmalloc-8 through kmalloc-256 slab caches. Qualys demonstrated successful root escalation on Debian 13.1 using AF_PACKET page vectors to leverage this condition into controlled kernel memory corruption [2].

Uncontrolled Recursion (Denial of Service / Kernel Panic). The `__remove_profile()` function processes nested AppArmor subprofiles recursively. On x86-64 systems, the 16 KB kernel stack is exhausted when processing profiles nested more than approximately 1,024 levels deep, triggering a kernel panic and forced reboot. An attacker with the ability to load profiles — including via the confused deputy path — can exploit this to cause denial of service on any affected system [2].

The remaining four vulnerabilities include an out-of-bounds read in the `unpack_pdb` profile unpacking path, a memory leak in the `verify_header` function, an out-of-bounds read and potential write in the `verify_dfa` function (rated as potentially enabling kernel code execution), and a race condition on rawdata structure dereference. Together with the five classes above, they present a multi-vector attack surface across AppArmor's policy management subsystem.

Container-Specific Impact

The security implications for containerized infrastructure extend well beyond the risks to traditional Linux hosts. In container deployments — including cloud-hosted Kubernetes clusters, container-as-a-service platforms, and on-premises container infrastructure — AppArmor profiles serve as a critical enforcement layer within a defense-in-depth stack. While namespaces, seccomp filtering, and capability restrictions provide partial isolation independently, AppArmor's role in restricting sensitive kernel features and syscalls means these vulnerabilities meaningfully degrade the isolation boundary even where other controls remain in place.

The confused deputy vulnerability is particularly dangerous in multi-tenant container environments. If an attacker can inject a malicious output redirection into a privileged host process — for example, through a misconfigured container volume mount, a supply chain compromise in a container image, or a compromised CI/CD pipeline that writes to shared host paths — they can load an AppArmor profile that removes restrictions from target container processes. This effectively disables the container's enforcement boundary without requiring any additional exploit or container escape technique.

Qualys and Canonical explicitly stated in their advisories that container-specific mitigations are insufficient as standalone responses. Because these are kernel-level vulnerabilities, any container runtime, orchestration platform, or application-level security control that relies on AppArmor enforcement inherits the full exposure. The only reliable remediation for container environments is kernel patching [1][5].

Affected Scope

The affected scope spans all Linux distributions with AppArmor enabled running kernel v4.11 or later. Ubuntu versions from 14.04 LTS through the current release (25.10 Questing Quokka [5]) are affected, with all nine vulnerabilities present in Ubuntu 22.04 LTS, 24.04 LTS, and 25.10. Earlier releases (16.04 and 14.04) are exposed to the denial-of-service variants. Debian 13.1 is confirmed exploitable for root escalation. SUSE Linux Enterprise and openSUSE, which enable AppArmor by default, fall within the exposure window. Distributions that use SELinux rather than AppArmor as their primary MAC enforcement mechanism — such as Red Hat Enterprise Linux, CentOS Stream, and Fedora — are not affected by these specific vulnerabilities [2][3][4].

The two-CVE assignment at time of disclosure is a procedural artifact: the Linux Kernel CVE Numbering Authority typically assigns identifiers in batches following stable kernel release, and the seven pending assignments are expected to arrive within one to two weeks. The absence of CVE numbers should not be interpreted as lower severity for the unassigned flaws [2].

Recommendations

Immediate Actions

Patching is the most reliable and fully effective remediation; no configuration-based mitigation eliminates all vulnerability classes in this set. The highest-priority action for any organization running AppArmor-enabled Linux systems — regardless of whether those systems run containers — is to apply vendor-supplied kernel security updates and reboot.

For Ubuntu environments, Canonical has released updated kernel packages for all supported releases. Teams should apply the updates through their standard patching channels using `apt update && apt upgrade` and confirm the installed kernel version reflects the patched release before rebooting. The following userspace components also require patching to harden the confused deputy attack path: `sudo` / `sudo-ldap` and `util-linux` (`su`). Both are available through Ubuntu's standard package repositories [5]. Organizations using `sudo-rs` as a replacement for the traditional `sudo` binary are not affected by the confused deputy chain, as the Canonical advisory explicitly confirms [5].

Qualys has published detection identifiers QID 386714 and QID 6032579 for organizations using the Qualys Vulnerability Management platform, enabling rapid identification of unpatched systems across an enterprise asset inventory [2].

Short-Term Mitigations

While patching is in progress, organizations should audit access controls on AppArmor's pseudo-filesystem entries. Restricting write access to `/sys/kernel/security/apparmor/.load`, `.replace`, and `.remove` beyond their default world-writable permissions removes the world-writable prerequisite for the confused deputy chain, reducing the attack surface for that specific vulnerability class, though the memory corruption vulnerabilities remain unaddressed by this control alone.

For Kubernetes and container orchestration environments, security teams should review which workloads are executing with host-path volume mounts, elevated capabilities, or security context exceptions that could provide an initial foothold for the confused deputy technique. Workloads that do not require such access should have those permissions revoked as a matter of defense in depth, independent of this vulnerability class. Runtime security tools that monitor kernel system call patterns — such as Falco — may provide early detection of unexpected AppArmor profile loading activity, though detection rule coverage for this specific technique should be validated before relying on it as a primary detection control.

Container images originating from external or third-party sources should be treated as potentially adversarial for the purpose of this vulnerability class. Qualys's exploitation chains for the use-after-free and double-free vulnerabilities are triggered entirely from within the context of the attacking process — a compromised container image executing with default permissions on an unpatched host has a viable path to kernel root escalation through these vulnerabilities.

Strategic Considerations

The CrackArmor disclosure is an instructive example of a vulnerability class that is frequently underappreciated in enterprise security programs: kernel security module implementation flaws. Organizations that have relied heavily on AppArmor as a defense-in-depth layer should reassess the degree to which that layer's correctness has been independently validated. The nine-year exposure window — from kernel v4.11 in 2017 through the March 2026 disclosure — suggests that critical MAC enforcement code does not receive the same continuous scrutiny as higher-profile kernel subsystems.

From a strategic posture perspective, this disclosure reinforces the importance of runtime monitoring at the kernel level. Host-based security tools that monitor kernel telemetry — including eBPF-based runtime security products and kernel audit subsystems — can detect exploitation attempts even when AppArmor itself has been subverted. Organizations that rely solely on AppArmor for MAC enforcement without independent runtime monitoring have a visibility gap that this vulnerability class can exploit.

For cloud-hosted infrastructure, teams should determine whether their cloud provider has patched the underlying host kernels for managed Kubernetes services (such as AWS EKS, Google GKE, and Azure AKS). Most major cloud providers have indicated patches are available or in rollout as of this writing, but organizations should verify node kernel versions directly through cluster management APIs rather than assuming managed service updates have been applied [3].

CSA Resource Alignment

The CrackArmor vulnerabilities sit at the intersection of several CSA research areas and control domains that provide contextual guidance for organizational response.

CSA Container Security Publications. CSA's *Best Practices for Implementing a Secure Application Container Architecture* (2019) provides foundational guidance on host security hardening and defense-in-depth principles for container deployments that are directly applicable to this threat. The document's emphasis on treating host hardening as a prerequisite for container security — including MAC enforcement, minimal capabilities, and runtime monitoring — reflects the layered posture that

CrackArmor demonstrates cannot be treated as fully trusted. CSA's companion publication *Building Incident-Resilient Containers* covers Linux security mechanisms including Seccomp, SELinux, and Linux capabilities alongside AppArmor, underscoring the importance of layered MAC enforcement rather than dependence on any single mechanism [6][7].

AI Controls Matrix (AICM). For organizations deploying AI inference workloads in containerized environments — a rapidly growing pattern across the enterprise — the AICM's infrastructure integrity controls apply directly. CrackArmor represents a vector through which an adversarial actor could compromise the host running an AI agent or model serving infrastructure, potentially enabling data exfiltration, model tampering, or agent behavior manipulation. The AICM's guidance on compute environment integrity and isolation requirements informs the baseline expectations that CrackArmor's disclosure shows must be continuously validated rather than assumed.

Cloud Controls Matrix (CCM) v4. CCM domain TVM (Threat and Vulnerability Management) provides the control framework for organizations to define policies around vulnerability prioritization, patch SLAs, and compensating controls. The multi-vector nature of the CrackArmor set — spanning denial of service, memory disclosure, and full root escalation — spans multiple CCM risk tiers and requires coordination between vulnerability management, platform engineering, and security operations teams. CCM domain SEF (Security Incident Management) applies to the response phase, particularly for organizations that need to assess whether exploitation occurred during the window between the March 12 disclosure and patch deployment.

MAESTRO Threat Model. The MAESTRO framework for agentic AI threat modeling identifies infrastructure compromise as a threat vector enabling agent subversion. CrackArmor is a concrete example of how a kernel-level flaw can bypass the isolation boundaries that agentic deployments depend on — an adversary with code execution inside an AI container on an unpatched host can potentially escape to the host kernel through this vulnerability set, undermining any agent-level trust boundary. Organizations deploying AI agents in containerized environments should factor kernel MAC enforcement integrity into their MAESTRO threat modeling exercises.

Zero Trust Architecture Guidance. CSA's Zero Trust guidance is relevant here in its treatment of workload identity and runtime integrity verification. A Zero Trust posture applied to compute workloads would include continuous verification of node and kernel integrity as a condition of trust, rather than treating a provisioned Kubernetes node as implicitly trusted after initial deployment. CrackArmor illustrates the practical consequence of that posture gap: a node with an unpatched kernel running AppArmor is operating with a known-compromisable isolation boundary, a state that a mature Zero Trust program would surface as a policy violation warranting immediate remediation.

References

[1] Qualys Threat Research Unit, "CrackArmor: Critical AppArmor Flaws Enable Local Privilege Escalation to Root," Qualys Blog, March 12, 2026. https://blog.qualys.com/vulnerabilities-threat-research/2026/03/12/crackarmor-critical-apparmor-flaws-enable-local-privilege-escalation-to-root

[2] Saeed Abbasi / Qualys TRU, "Multiple vulnerabilities in AppArmor," oss-security mailing list (Openwall), March 12, 2026. https://www.openwall.com/lists/oss-security/2026/03/12/7

[3] CSO Online, "Nine critical vulnerabilities in Linux AppArmor put over 12M enterprise systems at risk," March 2026. https://www.csoonline.com/article/4145539/nine-critical-vulnerabilities-in-linux-apparmor-put-over-12m-enterprise-systems-at-risk.html

[4] Infosecurity Magazine, "CrackArmor Flaws Expose Linux Systems to Privilege Escalation," March 2026. https://www.infosecurity-magazine.com/news/crackarmor-linux-privilege/

[5] Canonical / Ubuntu Security, "AppArmor vulnerability fixes available," Ubuntu Blog, March 2026. https://ubuntu.com/blog/apparmor-vulnerability-fixes-available

[6] Cloud Security Alliance, "Best Practices for Implementing a Secure Application Container Architecture," CSA Research, 2019. https://cloudsecurityalliance.org/research/artifacts/best-practices-for-implementing-a-secure-application-container-architecture

[7] Kaif Ahsan, Kumar Soorya, "From IF to WHEN: Building Incident Resilient Containers," Cloud Security Alliance, 2023. https://cloudsecurityalliance.org/research/artifacts/building-incident-resilient-containers

