CIQ

11 min read

The real danger of systemd-coredump CVE-2025-4598

August 15, 2025

Table of contents

INTRODUCTION
BACKGROUND
VULNERABILITY
AFFECTED SYSTEMS
SEVERITY
EXPLOITATION
PASSWORD CRACKING

DEFENSE IN DEPTH

TRADITIONAL PATCHING

Contributors

Solar Designer

Subscribe to our newsletter

Enter your email

SUBSCRIBE

TL;DR: A critical vulnerability in systemd-coredump remains unfixed in Enterprise Linux 9, allowing attackers to steal password hashes and cryptographic keys within seconds - but Rocky Linux from CIQ - Hardened (RLC-H) mitigates this:

- Attackers with basic system access can exploit this vulnerability to extract sensitive data (password hashes, crypto keys) from crashed privileged programs in seconds, not hours or days
- EL9 systems are vulnerable by default, while EL7/8 are not affected in default configuration - making this a significant risk for organizations running modern Enterprise Linux deployments
- The attack requires no special skills or complex setup - attackers can trigger crashes, manipulate process IDs instantly, and win timing races

reliably across different systems
- RLC-H blocks this attack through multiple defense layers: hardened sysctl settings, restricted SUID program access, and stronger password policy and hashing
- Live demonstrations show vanilla Rocky Linux 9.6 compromised in under 5 seconds with weak passwords cracked immediately

Bottom line: This isn't theoretical - it's a working exploit against unpatched EL9 systems.

Introduction

It's been over two months since the public disclosure of systemd-coredump CVE-2025-4598 by Qualys, yet as of this writing the vulnerability remains unfixed in upstream Enterprise Linux, and most importantly fully exposed by default on EL9. While RLC-H, which builds on EL9, has provided effective mitigations from the start, there is ongoing danger and potential implications of leaving this vulnerability for those without such protections, urging users and administrators to understand the risks and take necessary precautions.

The fact that Oracle also blogged about this CVE recently after having fixed it on the public disclosure date emphasizes the risk it presents. While Qualys and Oracle describe the vulnerability and its fixes in great detail, we demonstrate the vulnerability’s severity through its direct exploitation.

Background

When a running program crashes (or is made to crash), it may produce a so-called core dump containing the last state of the program's memory. This is intended primarily to let the program's or the distribution's developers or maintainers analyze the crash and fix any bugs that may have caused the

crash. Core dumps may contain whatever data the program was working on, including sensitive information.

While the Linux kernel would normally either write core dumps to properly protected files directly (which nevertheless has some risks) or not do it at all, it also supports a mode where core dump content would be redirected to a program. It's this mode that is used by many Linux distributions for centralized and distribution-specific processing of core dumps. Because of their non-trivial nature, the various core dump processing programs tend to contain vulnerabilities. Perhaps the first batch of those - in apport as used in Ubuntu and in abrt as used (back then) in Fedora and RHEL/CentOS - were discovered by Tavis Ormandy from Google in 2015. Then more vulnerabilities were discovered in apport by others, including later in 2015 and in 2017, 2019, 2020, 2021, and in systemd-coredump (which is part of the systemd package) as used in Fedora/RHEL/CentOS by Matthias Gerstner from SUSE in 2022. This year, it was apparently time for another round of vulnerability discovery in apport and systemd-coredump, this time by the Qualys Threat Research Unit (TRU).

Vulnerability

When systemd-coredump saves core dumps to files, it makes those readable to the user who was (presumably) running the crashed program. A problem with this is that a running (or now crashed) program's privileges are not always exactly those of the user, and a running program doesn't always have just one owner throughout its lifetime. Some program executables are marked in the filesystem with the SUID ("Set user ID on execution") or/and SGID (ditto for group ID) flags or/and with a set of so-called capabilities (privileges) that the invoking user may not have had but the program would. Some other programs (most notably some of the so-called daemons) may have started execution with great privileges (commonly as the root user) and switched to run as a relatively unprivileged user afterwards. These may retain some privileged access or/and remnants of data that the user couldn't access

directly. The kernel maintains a flag called "dumpable", which is correctly reset in those special cases preventing the kernel's usual creation of a user-readable core dump - but not redirection to a core dump processing program such as systemd-coredump.

Until 2022, systemd-coredump did not handle these special cases at all, so core dumps were made readable by whatever user the program appeared to be running as at time of crash. After the fix in 2022, systemd-coredump attempted to duplicate the kernel's logic in determining whether a core dump can safely be made readable to the user or not. In 2025, Qualys found that this logic may not always be looking at the actual crashed program as its process ID may have already been reused (a race condition). The new upstream systemd fix is to obtain and use a copy of the kernel's dumpable flag.

Affected systems

Although mainstream Linux distributions use systemd these days and systemd-coredump is part of the systemd package, so technically the vulnerability is present in the package, not all of these distributions and systems are actually affected. Other prerequisites for the vulnerability to be exposed are having systemd-coredump as the configured handler in the kernel.core_pattern sysctl (which systemd itself may configure, depending on its configuration) and the fs.suid_dumpable sysctl having a non-zero value. Most relevantly, these conditions are met by default on recent Fedora, EL9, and EL10, but are not on EL8 and EL7 / CentOS 7. Since Fedora has issued a systemd update with the fix promptly and EL10 is just starting to gain adoption, EL9 with its delayed fix and extensive adoption stands out as the most relevant target.

Severity

Overall risk severity is commonly measured in terms of a combination of the probability of occurrence of an event (risk probability) and its consequence (risk impact). The probability may further be combined from the likelihood of existence of the threat (such as attempted exploitation of a vulnerability) and that of the attack succeeding before the threat actor would give up (reliability of exploitation).

This particular vulnerability is exposed on the system "locally", which actually means it's subject to threats from anyone already able to run code on the system as an unprivileged user, including through remote access. Once they run pre-tested exploit code, we estimate that their probability of prompt successful exploitation on a new system is very high - more on this below.

The immediate consequence of exploiting this vulnerability is access to sensitive data, such as password hashes or cryptographic keys. Importantly, per the mostly overlooked "Last-minute update" paragraph in the Qualys advisory, sensitive data can be obtained not only from SUID programs and alike, but also from some daemon processes such as OpenSSH's sshd-session and systemd's own sd-pam. Although this only directly impacts confidentiality rather than integrity and availability, it does also impact those indirectly through use of password hashes to crack the actual passwords or through unauthorized use of cryptographic keys.

systemd upstream and Red Hat evaluated this vulnerability as having a CVSS v3.1 score of 4.7 (Medium) due to the vector CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:N/A:N. This may have been technically correct (or not) and it needs to be consistent with how other issues are rated by the same parties, but it ends up downplaying the severity of the issue now. Further, Red Hat rated this Moderate per their own 4-point scale. We suggest it should be Important instead.

Exploitation

There's a lot of detail on exploitation in the Qualys advisory and in the oss-security thread that followed, and there are pretty diagrams in the Oracle blog post, all referenced above. So we won't repeat this here. We will instead note a few important aspects that ease exploitation:

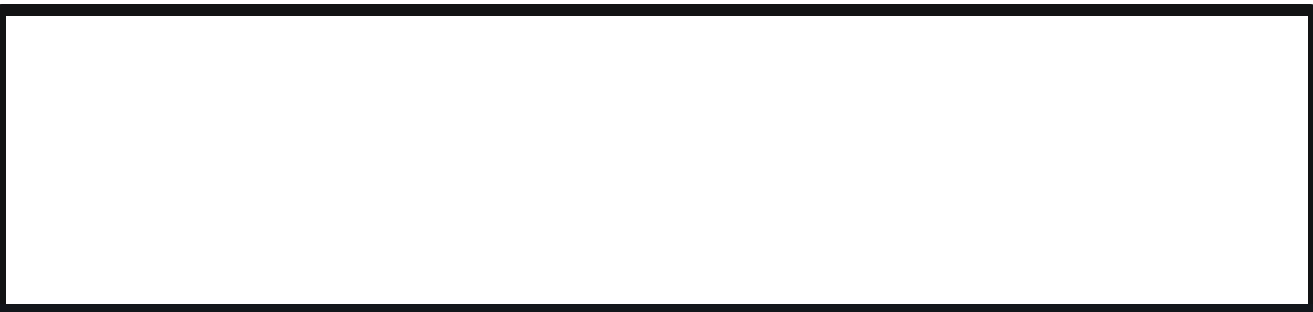
Crashing a program such that it would dump core may sound like it'd require finding a bug in the program first, but this is not the case. For a program that (partially) runs as the user (which is the special case we're targeting anyway), this is as easy as sending a certain signal to the process.

Triggering process ID (PID) reuse may sound like it'd take a long while given the typical kernel configuration with over 4 million PIDs, but as Vegard Nossum from Oracle pointed out the kernel has a (mis)feature allowing one to set the next PID directly. Further, the kernel fails to restrict this to true root but would also process such a request from a suitable SUID root confused deputy program, one of which is known - it is newgrp. While the primary shortcoming is in the kernel, for now Vegard got a workaround accepted into upstream newgrp code - but anyway the version in EL9 is older, so it lacks that protection. Thus, deliberate PID reuse on EL9 is actually instant.

Winning the race (having PID reuse happen at just the right time) may sound like it'd take a lot of trial and error, but in our experience once the exploit program is made to work reliably on a given system it tends to succeed almost or literally from the first try also on other systems running the same or similar Linux distribution, and that's even despite of typical variation of CPU clock rate, VM vs. bare metal, etc.

With all of the above combined, the attacker can expect to have e.g. password hashes within seconds. This includes root's password hash as seen when targeting the unix_chkpwd SUID root program.

Let's just do it as a proof of concept:



First we run the exploit on Rocky Linux 9.6. It wins the race within a second (as it happens, from the third try - but retries are automated and are very quick). It dumps password hashes that originally came from /etc/shadow, including (as it happens) 3 copies of the user's (including a full line from /etc/shadow with the username and other metadata) and 1 copy of root's (an almost full line from /etc/shadow, with only the initial 3 characters missing, so we see it start with "t:" instead of "root:").

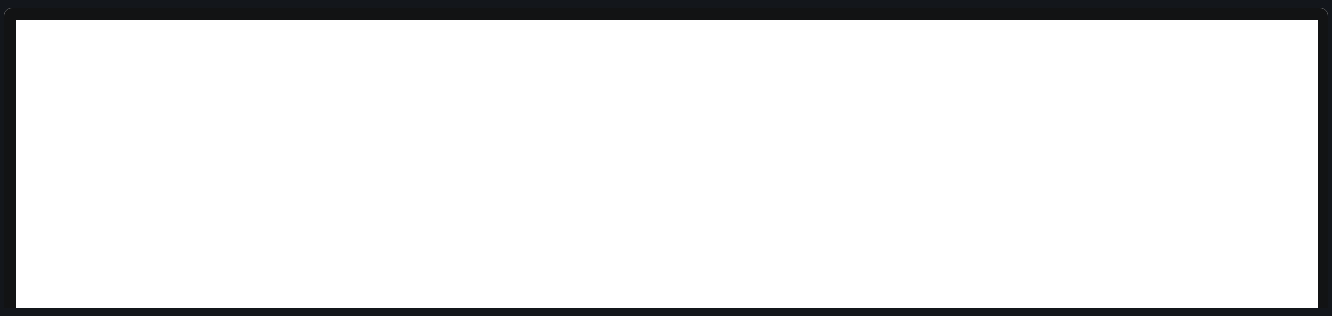
Then we run the same exploit on RLC-H 9.6. The exploit keeps failing due to RLC-H's default mitigations, so we interrupt it with Ctrl-C. We then undo 2 mitigations, deliberately weakening security: use our "control" framework to re-expose unprivileged access to the unix_chkpwd and newgrp programs, and change fs.suid_dumpable from RLC-H's safe default of 0 to 1 (2 would also work). With this, the exploit finally works (as it happens, succeeds from the first try). It also dumps password hashes with the same peculiarities as above (since we're targeting the same program, unix_chkpwd), but due to another change we made in RLC-H the hashes are of a different type.

Password cracking

Can the actual passwords realistically be inferred from the hashes? Luckily, all of these hashes are good enough that they cannot be reversed other than through testing candidate passwords against them. So our chances for success depend on how weak or strong the passwords are, and on how many guesses we can test before we'd give up.

Let's try with John the Ripper password cracking tool (latest git revision) on a rented VDS with one NVIDIA RTX 5090 GPU and 16 vCPUs from the larger AMD EPYC 9354 CPU (so this looks like 1/4th of the physical server), itself running Rocky Linux 9.6:

Demo of password hash cracking for vanilla Rocky vs. RLC-H hashes on CPU vs. GPU



Please note that delays in this recording have been capped at 1 second not to keep you waiting. Please look at the actual session durations as reported by the tool.

The hashes we got from Rocky Linux 9.6 are sha512crypt. We start by benchmarking sha512crypt at its historical default setting of 5000 rounds. The speeds are either 63k per second on the vCPUs (on all 16 of them, and using AVX-512) or 1640k per second on the GPU (26 times faster). However, EL 9.5+ upgraded the rounds setting it uses for sha512crypt from 5000 to 100000 (so by a factor of 20), so that's what our actual target hashes use. We try to crack them, first on the CPU, which shows a speed of 3150 combinations (of candidate password and target hash) per second (20x slower than our initial benchmark, as expected).

We succeed in cracking both in 5 seconds: the passwords turn out to be secret666 for user and fullaccess for root. These are very weak passwords that are within top 10k in John the Ripper's default password.lst file, which is a list of common passwords ordered for decreasing number of occurrences in breaches. Yet both of these passwords, as well as 42 more from the top 10k, are accepted by the default pwquality password policy on EL 9.6.

Then we try the same on the GPU. It also cracks the passwords, but takes 34 seconds to do so. That's because GPUs are best at larger jobs, such as testing a far larger number of candidate passwords at once. If you're familiar and look closely, the tool actually says its auto-tuned global work size is over 1 million, so that's how many candidate passwords it tests in parallel, which for weak passwords from the top 10k is unhelpful. This would be great for more complex passwords, and it actually shows a speed of over 81k combinations per second (26x faster than the vCPUs, as expected). However, for passwords so weak, we can do a better job by limiting it to testing fewer in parallel. When we do, it cracks both in under 1 second (albeit at a non-optimal speed in terms of combinations per second).

Then we move to the hashes we got from RLC-H, which are yescrypt. A benchmark shows a little over 1500 per second on the vCPUs. We then try to crack the actual hashes, which goes on at the same kind of speed for a while without success, and we interrupt. The very weak passwords we found above wouldn't have been accepted by RLC-H's default passwdqc password policy - in fact, none from the top 10k would be. yescrypt is not currently implemented on GPU and it is deliberately GPU-unfriendly by design, so by far not as much speedup is expected as we saw for sha512crypt. With the current defaults and currently available code, it is ~50 times slower to crack on this system than 9.6 default sha512crypt (1.5k vs. 81k), and that's on top of EL's recent upgrade of the sha512crypt rounds.

Defense in depth

We just saw how an unfixed vulnerability poses very real danger when defense-in-depth is lacking, and is mitigated otherwise. While RLC-H's change of default for fs.suid_dumpable fully prevents the vulnerability, its several other changes (restricted access to dangerous SUID root programs, upgraded password policy and password hashing) would have partially mitigated the vulnerability even if this main change were not present.

While we do recommend RLC-H as our supported product, we also contribute these individual enhancements to the Rocky Linux community via SIG/Security, where they can be used for DIY hardened setups based on community Rocky Linux or even on top of other Enterprise Linux distributions.

Another RLC-H and SIG/Security component that can fully prevent this vulnerability is Linux Kernel Runtime Guard (LKRG), albeit currently only in non-default configuration: its sysctl settings lkrg.umh_validate=2 or lkrg.profile_validate=4 (so-called paranoid mode) completely block Linux kernel's "usermodehelper" feature, and thus would block invocation of systemd-coredump by the kernel. We do not enable this by default because it'd also block dynamic loading of drivers, including on system bootup. We may make this setting more fine-grained in a later update, so that we'd enable just the desired defense by default.

We're also planning even further improvements to passwdqc password policy and yescrypt settings.

Traditional patching

While not the focus of this post, we do indeed expect to have this CVE patched also in the traditional manner, via our respected upstream and Rocky Linux. Separately, we're patching it for our 9.x LTS products and (as a rare exception since the delay otherwise became inappropriate) via the base RLC 9 product's FastTrack repository.

1.4M+

Rocky Linux instances

Being used world wide

90%

Of fortune 100 companies

Use CIQ supported technologies

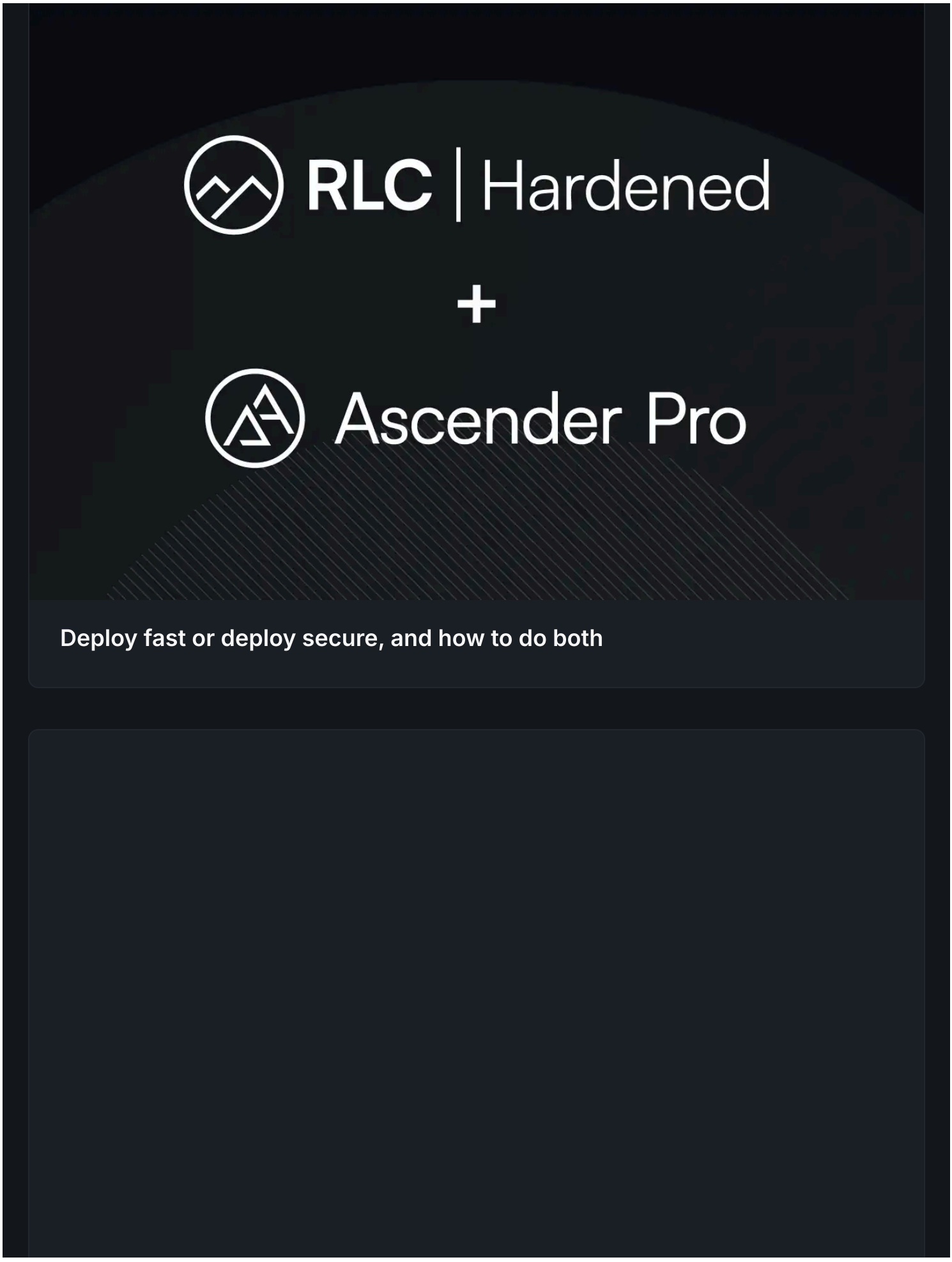
250k

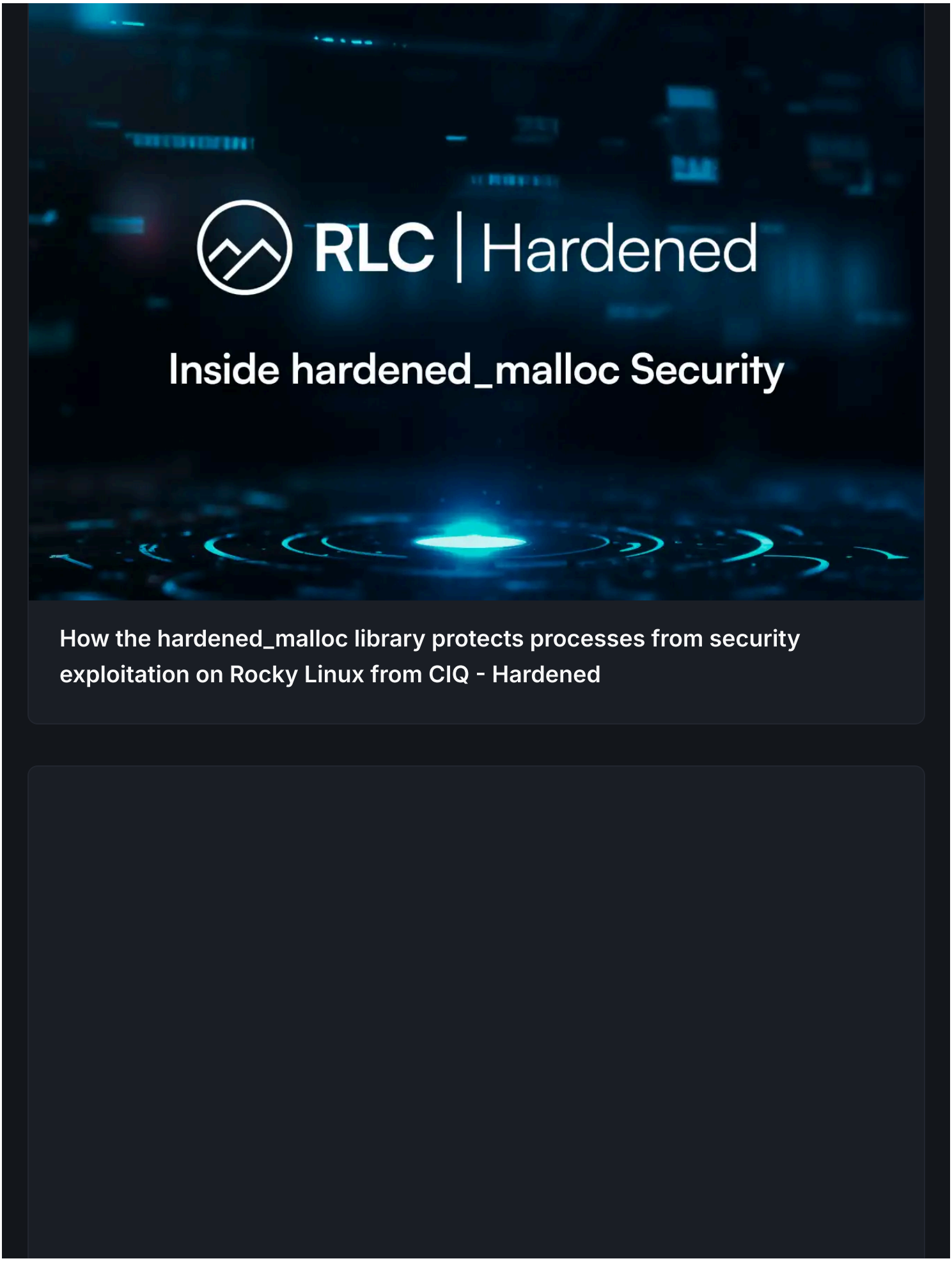
Avg. monthly downloads

Rocky Linux

Related posts

Available Now: A Security focused Linux... and pre-configured compliance options


RLC | Hardened
+
Ascender Pro
Deploy fast or deploy secure, and how to do both



How the hardened_malloc library protects processes from security exploitation on Rocky Linux from CIQ - Hardened

Linux kernel CVEs 2025: what security leaders need to know to prepare for 2026

CIQ

Contact

(800) 220-5243

INFO@CIQ.COM

Products

ROCKY LINUX

ASCENDER PRO

FUZZBALL

WAREWULF PRO

APPTAINER

CIQ BRIDGE

Support

LIFECYCLE SUPPORT

LONG-TERM SUPPORT

CONSULTING

MIGRATION

LOGIN

KNOWLEDGE BASE

Learn

BLOG New Post

GLOSSARY

RESOURCE LIBRARY

TRUST CENTER

About

ABOUT CIQ

FOUNDING STORY

OPEN SOURCE ETHOS

TEAM

NEWSROOM

FAQS

CAREERS We're hiring!