

How Dirty Frag rose from the Linux exploit Copy Fail

reversinglabs.com/blog/dirtyfrag-linux-privilege-escalation-exploit

Igor Lasic

May 12, 2026

How Dirty Frag rose from Copy Fail exploit

RL documented 163 samples of the Linux exploit's new variants, active malware — and developed YARA rules.



ReversingLabs (RL) researchers analyzed CVE-2026-31431, a Linux kernel local privilege escalation vulnerability tracked in public reporting under the names Dirty Frag and Copy Fail (CVE-2026-31431). The public embargo was broken in early May 2026 with press coverage citing the vulnerability across all major Linux distributions; Ubuntu released fixes shortly after. RL's corpus reveals that classified malicious samples were already in circulation at least 9 days before the embargo break, with sustained sample arrival continuing through publication.

As of May 8, 2026, RL has identified 163 unique samples tied to this vulnerability across two parallel threat-name conventions: 148 samples carry the *exploit:CVE-2026-31431* tag with names such as *Linux.Exploit.CVE-2026-31431*, *Linux.Exploit.CopyFail*, and *Linux.Trojan.Multiverze*, and an additional 15 samples derived from the V4bel/dirtyfrag GitHub reference implementation are tracked separately under *Linux.Exploit.DirtyFrag* and *Linux.Trojan.DirtyFrag*. Sample variants span ELF binaries, Python scripts, and a malicious PyPI wheel.

This post documents the sample landscape, explains the shellcode analysis methodology applied to the V4bel reference implementation, and provides YARA rules and hunting queries that defenders can deploy immediately to identify CVE-2026-31431 exploit code across all observed variants.

Dirty Frag threat background

CVE-2026-31431 is a Linux kernel privilege escalation vulnerability. Press coverage characterized it as Copy-Fail-like, referencing the Dirty Pipe class of vulnerabilities (CVE-2022-0847) that abuse the kernel's page-cache to write content to read-only memory regions without the required permissions. The structure of the public reference exploits is consistent with that architectural domain. The RL corpus tracks samples under two parallel threat-name conventions reflecting how the vulnerability has been referenced in different research streams: Copy Fail (the earlier press nickname), CVE-2026-31431 (the official designation), and Dirty Frag (the name used by the V4bel/dirtyfrag GitHub reference implementation).

The earliest sample classified malicious as *Linux.Exploit.CVE-2026-31431* was first seen on April 29, 2026 at 21:19 UTC. A substantial sample surge began on May 1, 2026, with more than 50 distinct samples submitted in a single overnight window. Sample arrival continued steadily through May 8, when the V4bel/dirtyfrag reference implementation was published on GitHub and circulated in press coverage citing the broken embargo.

Sample Dirty Frag landscape

The 148 samples tagged with the CVE-2026-31431 exploit field decompose into the following threat-name distribution. The 15 V4bel-derived samples are listed at the bottom under their separate threat-name set.

Opcode bytes	Assembly	Syscall	Function
b0 6a 0f 05	mov al, 0x6a; syscall	106	setgid(0) — set group ID to root
b0 69 0f 05	mov al, 0x69; syscall	105	setuid(0) — set user ID to root
b0 74 0f 05	mov al, 0x74; syscall	116	setgroups(0, NULL) — clear supplementary groups

The fourth string targets the `execve` call. Here the shellcode uses `push 0x3b; pop rax` to load syscall 59 (0x3b) without embedding a null byte. This is standard position-independent shellcode tradecraft for `/bin/sh` invocation.

Condition logic

Both rules require all of them: every opcode pattern plus plaintext strings `/bin/sh` and `TERM=xterm`. The `TERM=xterm` environment variable is written by the shellcode to produce a functional interactive terminal in the spawned shell. Its co-occurrence with the privilege normalization syscall patterns is a reliable indicator of the reference payload.

Coverage scope

These YARA rules target the V4bel reference implementation and its derivative compilations. They will not match Python-script variants, which require text-based detection patterns rather than opcode patterns. They will not match the Multiverze trojan family unless it incorporates the same shellcode stub. For full corpus coverage, combine the YARA rules with the Spectra Intelligence threat-name and exploit-field hunting queries documented below.

YARA rule

```
rule DirtyFrag_Reference_Shellcode_1
{
    meta:
        author = "Malware Utkonos"
        date = "2026-05-08"
        description = "Detects shellcode from reference implementation of DirtyFrag"
        reference = "https://github.com/V4bel/dirtyfrag"
    strings:
        $op1 = { b06a 0f05 }
            // 0040007e b06a      mov     al, 0x6a
            // 00400080 0f05      syscall
        $op2 = { b069 0f05 }
            // 00400082 b069      mov     al, 0x69
            // 00400084 0f05      syscall
        $op3 = { b074 0f05 }
            // 00400086 b074      mov     al, 0x74
            // 00400088 0f05      syscall
        $op4 = { 6a3b 58 0f05 }
            // 004000a0 6a3b      push    0x3b
            // 004000a2 58        pop     rax    {0x3b}
            // 004000a3 0f05      syscall
        $a1 = "/bin/sh"
        $a2 = "TERM=xterm"
    condition:
        all of them
}
```

Figure 1. [PLACEHOLDER: Disassembly from Spectra Analyze showing the setgid/setuid/setgroups/execve syscall chain at addresses 0x40007e–0x4000a3 in the reference binary. Note the mov al compact pattern for syscalls 105, 106, and 116, and the push 0x3b; pop rax technique for execve.]

Hunting queries

The CVE-2026-31431 corpus is partitioned across two threat-name conventions in the RL classification system. To achieve full coverage, run all four queries. The first query is the strongest single hunt: it covers all 148 samples carrying the CVE-tagged exploit field, regardless of threat-name variant.

Spectra Intelligence: hunt by CVE exploit field (recommended primary hunt).
exploit:CVE-2026-31431

Spectra Intelligence: hunt by V4bel-specific threat names.
threatname:DirtyFrag

Spectra Intelligence: hunt for active trojan family adoption.
threatname:Linux.Trojan.Multiverze AND exploit:CVE-2026-31431

Spectra Intelligence: hunt for Python-script and supply chain variants.
threatname:Script-Python.Exploit.CVE-2026-31431
threatname:Package.Exploit.CopyFail
threatname:Linux.Exploit.CopyFail

Indicators of Compromise (IOC)

The full corpus contains 163 samples and is too large for a static table. The 25 entries below are a representative subset covering the V4bel reference implementation, the highest-detection samples, the Multiverze trojan family, the Python and PyPI variants, the Copy Fail-named pre-CVE samples, and the earliest-observed samples in the corpus. The complete sample set is retrievable in real time via the hunting queries in the previous section. Hashes are SHA-256. AV column shows scanner detection count at time of writing.

SHA-256	Type	Threat name	AV	First seen (UTC)
e7fb35c16fbe6285d4f36764fe5f6f81b0ff51c047f5716bbb8ae60b8318d82e	ELF64/SO	Linux.Exploit.DirtyFrag	2	2026-05-08 13:27:52
133a79e9094c14c0f41378c712fd9a3f7687e5ab6f781bd5fb94774e64f4b48d	ELF64/SO	Linux.Exploit.DirtyFrag	2	2026-05-07 23:37:10
381755b623dd7a4c2b5d80aaf40d7083eea727dd1f473545539029656ca81817	ELF64/SO	Linux.Trojan.DirtyFrag	4	2026-05-08 11:01:45
a02ea2ba8108a9b7a997faa8808cfc55bb69af54e69178fa5aa1785681cf0ced	ELF64/Exe	Linux.Trojan.DirtyFrag, Linux.Exploit.DirtyFrag	4	2026-05-08 11:09:07
d99c480661fde92c3c7d1790c2e1d695fd72f4b82d47adb6e10093fd096c0708	Script/Python	Linux.Exploit.CVE-2026-31431	17	2026-05-05 09:13:05
48c0bb0760a08a70fa6cf96c0102c968cb1bc62d319cba0a605247be1e2e4180	ELF64	Linux.Exploit.CVE-2026-31431	15	2026-05-05 10:43:12

76ad71ac3cf6d50bf4038048b9832df5e9aa63b85865c02ad1dd91cb2fdaef4b	ELF64	Linux.Exploit.CVE-2026-31431	13	2026-05-05 16:46:10
ea21dbc2c11ee666cb9e2b4d2cd1e6a4776b3ea6bff6d57f80a6cf31624791e9	ELF64	Linux.Exploit.CVE-2026-31431	13	2026-05-01 06:33:38
bd855eb0a90c8cb6618662c48cc93d3a16cf9a7e4d945b70e3be3500f60042f9	Script/Python	Linux.Exploit.CVE-2026-31431	12	2026-05-01 17:57:55
26865ea1744e00664a13b1a65f2e670def8d3bb84b10533f18f2e0ac43548fe0	ELF64	Linux.Exploit.CVE-2026-31431	12	2026-05-07 03:12:51
912714027c9ea12b8aac55d71ccfa4a0592e058a4d07cf578e67f4bfdab63c4a	ELF64	Linux.Trojan.Multiverze	7	2026-05-05 19:46:28
b090751120d4814744c24253a820a67db5c3b2957c0334cf7d52e7847d6af409	ELF64	Linux.Trojan.Multiverze	10	2026-05-02 21:20:03
d658fd3b2fe203180e6a3ef6863a5eb3cdd92cfecbaa68de5b8f550702762eab	ELF64	Linux.Trojan.Multiverze	9	2026-05-01 20:33:31
5bd7df1c89cf9f69e6003d73a8e3b9eab9cf6025e6911f0fec0451da1673d6f0	ELF64	Linux.Trojan.Multiverze	9	2026-05-01 20:37:11
c60c5527e49c3bb502b672b7fffc3eb5b8c6ff9d4e30eae5d559b54d2ce03a01	Script/Python	Script-Python.Exploit.CVE-2026-31431	8	2026-05-07 12:42:18
c935a349a974ef605b5a12141934d966315c0da5fe2343750815927a39f92881	Script/Python	Script-Python.Exploit.CVE-2026-31431	9	2026-05-06 08:33:56
affde15382361e2fb87a7d32a5260ab72cc5d2d734fd7de6d21a1c94d0f58d22	Script/Python	Script-Python.Exploit.CVE-2026-31431	11	2026-05-05 16:07:12
424d306e8cba73ce83af5faf051a169d957a10213509d7132b620f427b4159bb	ELF64	Linux.Exploit.CopyFail	11	2026-05-05 09:27:01
1507e6e6945bfdf652ef7ed2fe10e01245074fd54d29d8eca98f265a91c88e63	ELF64	Linux.Exploit.CopyFail	10	2026-05-03 10:42:00
7bd2a8093d38e2694199490642e91965bdc666121070330c76ae155b6581ce75	PyPI Wheel	Package.Exploit.CopyFail	7	2026-05-01 07:52:00
26a75e5ef8d30ae678596fafa56e1f191d17fd9a438c463cd7dcefb765c2fb94	Archive/ZIP	Linux.Exploit.CVE-2026-31431	2	2026-05-08 10:56:03

a567d09b15f6e4440e70c9f2aa8edec8ed59f53301952df05c719aa3911687f9	ELF64	Linux.Exploit.CVE-2026-31431	9	2026-04-29 21:19:51
3c5ec61632d0699e048d8428461c4d65f89988a370396db2f070f63ebbf9dbae	ELF64	Linux.Exploit.CVE-2026-31431	11	2026-04-30 03:09:57
d401e7d1c00605749d6c617ace73ab20a762b72e41c2e1590331596e38219a61	ELF64	Linux.Exploit.CVE-2026-31431	10	2026-04-30 09:52:21
ee2d150a2f73a561983088a6b1a6a2b1c452777aaf03181387708c6907ac6dcd	Text/CCPP	(goodware - source code)	0	2026-05-07 23:30:56

MITRE ATT&CK mapping

ID	Technique	Sub-technique	Relevance
T1068	Exploitation for Privilege Escalation	—	Core exploitation: CVE-2026-31431 kernel vulnerability triggers root access
T1548	Abuse Elevation Control Mechanism	.001 Setuid and Setgid	Shellcode calls setuid(0), setgid(0), setgroups(0) to normalize root credentials
T1059	Command and Scripting Interpreter	.004 Unix Shell, .006 Python	/bin/sh spawned via execve; Python-script variants observed in the corpus
T1195	Supply Chain Compromise	.002 Compromise Software Supply Chain	Malicious PyPI wheel (copyfail-0.1.0-py3-none-any.whl) carries CVE-2026-31431 exploit code

How ReversingLabs can help

After uploading the YARA rules to Spectra Analyze, continuous local matching begins immediately. Every new Linux ELF file submitted to the appliance is evaluated against *RL_DirtyFrag_Linux_PrivEsc_Shellcode_2026* and matches surface in the YARA Hunting results without manual intervention. Also upload the ruleset to Spectra Intelligence to extend matching to the Spectra Intelligence cloud sample set as new files arrive. Cloud Retro Hunting runs the ruleset against RL's cloud sample set, covering the full window of CVE-2026-31431 sample arrival from April 29, 2026 onward.

For Python-script and PyPI-package variants that the YARA rules do not cover, Spectra Intelligence threat-name and exploit-field queries provide complete corpus visibility. The *exploit:CVE-2026-31431* query alone returns 148 samples spanning every variant type observed in the wild.

For software supply chain exposure, Spectra Assure and the free [Spectra Assure Community](#) analyze open-source package risk. The malicious *copyfail* PyPI wheel identified in this research demonstrates the exposure: a transitive dependency in any Python build pipeline could pull a kernel exploit into the developer environment. Scan your inbound package set for this hash and family pattern.

At the time of writing, AV detection rates across the CVE-2026-31431 corpus range from two to 17 scanner matches. RL's analyst classification and Spectra Core complex binary analysis confirmed malicious classification across all 163 samples ahead of broad AV consensus. This is the operational value of deploying these queries and rules now: you are not waiting for vendor signature updates.

Key recommendations for remediating Dirty Frag

- **Patch immediately.** Ubuntu has released fixes for CVE-2026-31431. All Linux distributions are reported affected. Apply vendor kernel updates and reboot to load the patched kernel. See vendor advisories for Red Hat, Debian, SUSE, and other distributions.
- **Hunt with the CVE exploit field as the primary query.** *exploit:CVE-2026-31431* returns the full set of 148 samples carrying this CVE in the RL corpus, regardless of which variant or threat-name convention applies. Run this first.

- **Add the V4bel-specific queries.** *threatname:Linux.Exploit.DirtyFrag* and *threatname:Linux.Trojan.DirtyFrag* cover the 15 reference-implementation samples that do not carry the CVE exploit field. Run both.
- **Deploy the YARA rules now.** Upload both rules to Spectra Analyze and enable continuous cloud matching. Start a retro hunt to cover files submitted in the past, which spans the full pre- and post-embargo window.
- **Scan your software supply chain for the malicious PyPI package.** The *copyfail-0.1.0-py3-none-any.whl* wheel (SHA-256 *7bd2a8093d...*) is in PyPI distribution scope. Use Spectra Assure to scan inbound packages and their transitive dependencies in your build pipelines.
- **Investigate Multiverse trojan presence.** *Linux.Trojan.Multiverse* samples in the corpus indicate active malware family adoption of this exploit. Hunt for multiverse indicators in your environment, particularly on hosts that have not yet received the kernel patch.
- **Audit *setuid* and *setgid* binaries on production Linux hosts.** Any binary with the *setuid* bit set is a potential post-exploitation target. Review the list and remove permissions where not required.
- **Alert on ELF execution from writable paths.** Monitor for execution of ELF binaries staged in */tmp*, */dev/shm*, or world-writable directories. These are common staging locations for locally compiled exploit code.
- **Monitor for privilege escalation process chains.** Alert on process events where a low-privilege parent spawns a child process with UID 0, particularly where the parent binary is an unsigned ELF or a Python interpreter executing a script not in your software inventory.

The detection gap is now

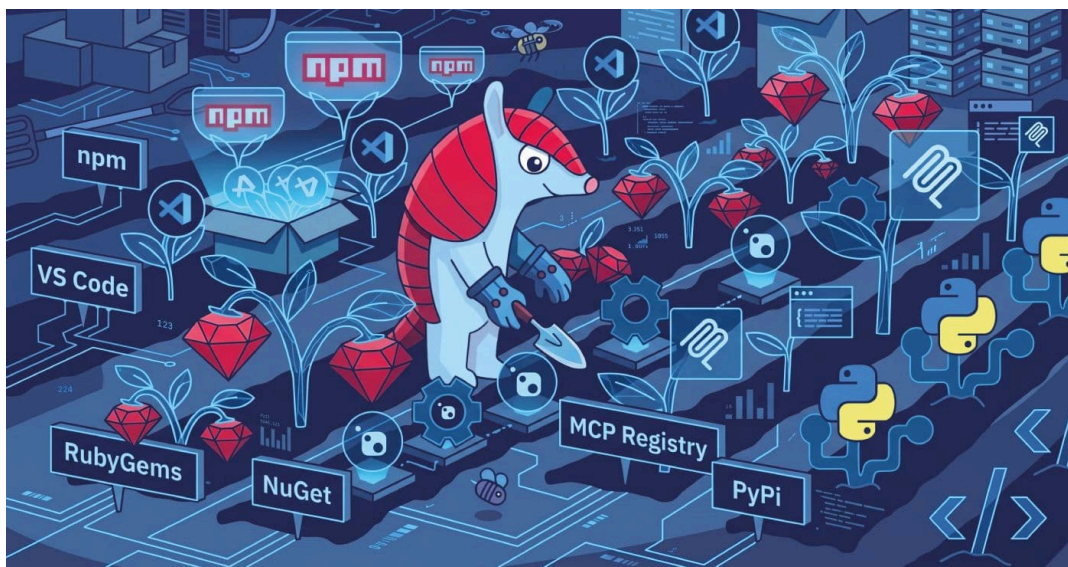
CVE-2026-31431 is broader, older, and more weaponized than the public embargo break narrative suggests. RL classified the first malicious samples on April 29, more than a week before the public embargo break drew press attention. By May 8, the corpus contained 163 samples spanning compiled ELF binaries, Python scripts, a PyPI wheel, and active trojan family adoption by *Linux.Trojan.Multiverse*. The *V4bel/dirtyfrag* GitHub reference implementation, often treated in coverage as the exploit, is one of multiple parallel research and development streams.

The shellcode pattern in the V4bel reference implementation is structurally stable, and the YARA rules documented here detect it with high specificity. RL researchers assess with high confidence that the rules detect the V4bel reference binary without modification. The team also assesses with moderate confidence that they will match derivative compilations preserving the shellcode stub. We assess with low confidence that the rules will materially affect the Multiverse, Python-script, or PyPI-package variants, which require text-based and metadata-based detection rather than opcode pattern matching. Combine the YARA rules with the Spectra Intelligence hunting queries to achieve full corpus coverage.

Patch the kernel. Run the queries. Deploy the rules. The detection gap is now.

Tags: [Threat Research](#)

Topics



Why RL Built Spectra Assure Community

We set out to help dev and AppSec teams secure the village: OSS dependencies, malware, more. Learn how.

[Read More about Why RL Built Spectra Assure Community.](#)
[Why RL Built Spectra Assure Community.](#)

SUBSCRIBE

Get the best of RL Blog delivered to your in-box weekly. Stay up to date on key trends, analysis and best practices across threat intelligence and software supply chain security.