

CVE-2026-98641

Microsoft bpf, arm64: set up the frame pointer for the exception callback

In the Linux kernel, the following vulnerability has been resolved: bpf, arm64: set up the frame pointer for the exception callback A program acting as exception boundary saves all callee-saved registers, so build_prologue() takes the exception_cb path and never calls push_callee_regs(). That is the only place find_used_callee_regs() runs, and with it the only place ctx->fp_used is set, so the callback prologue does not emit the mov x25, sp that points BPF_REG_FP at the frame the callback runs on.

EVRS 0 /100 Informational priority	CVSS — Not scored	EPSS 0.002 5.7th percentile · FIRST	CISA DEADLINE — No CISA deadline
---	---------------------------------------	---	--

Signals behind the score

☒ counted in the score ☐ not counted

evidence for 2 of 8 signals

☐ CISA KEY	Not currently CISA KEY-listed	☐ Vendor exploitation	Active-exploitation status not yet checked - scored at the population base rate (none) until known
☐ Ransomware	Ransomware association unavailable	☐ KEV recency	CISA KEY listing date unavailable
☐ EPSS	0.002 probability	☐ Pre-patch disclosure	Disclosure status unavailable
☐ CVSS	CVSS base score unavailable	☐ Microsoft Exploitability Index	Not a Microsoft CVE

Remediation

Monitor Untagged reference available; patch status is not confirmed

NVD lists one or more references for this CVE that aren't tagged as a patch or vendor advisory. Review them directly to confirm patch availability and applicability before making a remediation decision.

Basis: NVD reference (untagged)

x25 keeps whatever it held when bpf_throw() was called. If the throw came from a subprogram that uses its own BPF stack, that is the subprogram's frame pointer, and since the subprogram never returns it never restores x25 either. Stack accesses through BPF_REG_FP are rewritten to be stack pointer relative, so those still land in the callback's own frame. Materializing the register does not: a callback that passes the address of a local variable to a helper hands over an address in the dead subprogram's frame. That address is below the callback's stack pointer by then, and the helper's own call chain covers it, so the helper can write over its own return address. 0x1234 below is the value the helper was asked to store: pc : 0x1234 lr : 0x1234 Call trace: 0x1234 (P) bpf_test_run+0x188/0x3e0 bpf_prog_test_run_skb+0x47c/0x998 __sys_bpf+0xbdc/0xdd8 Kernel panic - not syncing: Oops: Fatal exception in interrupt Set ctx->fp_used on the exception callback path so that the existing code further down sets x25 from the stack pointer. The epilogue restores it from the main program's save area along with the other callee-saved registers, as it already does. x86 sets the frame pointer for the callback from the argument it is passed, and powerpc computes it from the stack pointer.



Affected and fixed versions

PACKAGE	AFFECTED	FIXED IN	SOURCE
Not identified by the connected sources	–	no fixed version listed	NVD

Sources

Reference	git.kernel.org/stable/c/b67b4c763f211e7a6391c356fef552e2ea63f08a	Additional reference	git.kernel.org/stable/c/d7cd5ebe13989f265015910faa25e5d50faa4c22
Additional reference	git.kernel.org/stable/c/ecdd770a9fc3eaf6c3127c5991312b84774a7b0f	Additional reference	git.kernel.org/stable/c/ef1fb82f12186dd26153b14d9fbcf4ec98db81b3
NVD	nvd.nist.gov/vuln/detail/CVE-2026-98641	ENISA EUVD	euvd.enisa.europa.eu/vulnerability/EUVD-2026-103555
Current data	evulnable.com/vulnerability-management/cve/CVE-2026-98641		

Snapshot at generation time. EPSS can move quickly. A public exploit module is not evidence of active exploitation.