I. Introduction

In August 2025, Apple and WhatsApp released emergency advisories for CVE-2025-43300 and CVE-2025-55177—a pair of vulnerabilities that, chained together, had already been used in the wild to compromise the mobile devices of fewer than 200 users over the preceding 90 days [**?**], [**?**]. Independent reporting identified many of the targets as journalists, human-rights defenders, and civil-society figures. The chain required no clicks, no downloads, and no error on the victim's part: receiving a WhatsApp message was sufficient. Both CVEs were added within weeks to the U.S. Cybersecurity and Infrastructure Security Agency (CISA) Known Exploited Vulnerabilities (KEV) catalog [**?**], mandating remediation across federal networks.

Zero-click chains of this shape have become the defining offensive capability of commercial spyware. Unlike phishing, which relies on social engineering, and unlike drive-by compromise, which relies on the victim visiting an attacker-controlled URL, a zero-click chain weaponizes the infrastructure that modern mobile operating systems use to render inbound media. Knowledge of the victim's phone number is sufficient; no further action on the victim's part is required. The asymmetry of this attack model—bounded defender effort against unbounded attacker opportunity—makes zero-click delivery the most consequential primitive in targeted surveillance today [**?**].

What makes the 2025 chain technically striking is its genealogy. Both vulnerabilities fire in code written in C—a language first standardized in 1989 whose memory model (Section **??**) provides no enforcement of bounds, type safety, or lifetimes. The exploitation stage is a heap out-of-bounds write in an image parser, a bug class first formalized by Aleph One in 1996 [**?**] and still ranked first in the MITRE CWE Top 25 list three decades later [**?**]. Despite repeated calls from CISA and the U.S. Office of the National Cyber Director for migration to memory-safe languages [**?**], the media-parsing frameworks at the core of every major mobile platform remain overwhelmingly C and C++. The result is a structural mismatch: a half-century-old memory model is entrusted with the first parsing step of every image delivered to every networked device.

This is not the first time the pattern has produced high-profile civilian casualties. In 2021, Citizen Lab and Google Project Zero reverse-engineered FORCEDENTRY (CVE-2021-30860), an NSO Group Pegasus zero-click chain with nearly identical architecture—iMessage delivery, CoreGraphics image parsing, heap corruption [**?**], [**?**]. Four years later, the 2025 WhatsApp–ImageIO chain reproduces the same three-step pattern in a different ecosystem, a different messenger, a different image format, and a different parser, with the same outcome. This recurrence across independent platforms and independent vendor teams is the central empirical observation motivating this study.

Research question. This paper asks: *how do decades-old memory-corruption patterns in C enable modern zero-click exploit chains, and what does the 2025 WhatsApp–ImageIO attack reveal about the persistence of these vulnerabilities in industry-critical software?*

Contributions. This paper makes three contributions:

1) A reconstruction of the 2025 zero-click chain from public primary sources—vendor advisories, patch diffs, and independent reverse engineering [**?**]—covering the delivery primitive (CVE-2025-55177, CWE-863) and the exploitation primitive (CVE-2025-43300, CWE-787), and showing how the two stages compose into a complete, interaction-free compromise.
2) A pedagogical bridge from the Exploit Education Phoenix `heap-two` exercise to CVE-2025-43300, demonstrating that the core mechanic—a heap buffer overflowed into adjacent function-pointer-bearing data—is the same primitive at two drastically different complexity levels. The bridge is intended for readers whose prior exposure to memory corruption stops at textbook stack smashing.
3) A structural argument that the FORCEDENTRY → WhatsApp–ImageIO lineage, spanning four years and two independent ecosystems, is evidence that the current mitigation stack (ASLR, sandboxing, pointer authentication) has not closed the fundamental attack class—only raised its price.

Scope and non-goals. This paper does not present a new exploit, a new vulnerability, or proprietary reverse engineering. All technical material is drawn from publicly disclosed sources. The Phoenix `heap-two` exercise is treated as a controlled pedagogical artifact, not a substitute for the real chain.

Roadmap. Section **??** introduces the C memory model, the two CWE classes at play, the iOS media pipeline, and the FORCEDENTRY precedent. Section **??** describes our CVE selection criteria, analysis framework, and Phoenix setup. Section **??** is the core technical analysis of the 2025 chain. Section **??** reflects on why the current mitigation stack fails to close this attack class. Section **??** concludes.