Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Memory-Safe eBPF/GPU Data Plane for Adaptive 5G Backdoor Detection

Michail Alexandros Kourtis

akis.kourtis@iit.demokritos.gr

National Centre of Scientific Research "Demokritos"

George Xilouris

National Centre of Scientific Research "Demokritos"

Research Article

Keywords:

Posted Date: November 5th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11389859/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Memory-Safe eBPF/GPU Data Plane for Adaptive 5G Backdoor Detection

Michail-Alexandros Kourtis · George Xilouris

Received: date / Accepted: date

Abstract The evolution of 5G towards cloud-native deployments exposes the User Plane Function (UPF) to persistent host-resident implants, of which the BPFDoor backdoor is a documented telecommunications-sector example. This work presents a tiered, memory-safe detection data plane that couples in-kernel extended Berkeley Packet Filter (eBPF/XDP) filtering with a memory-safe Graphics Processing Unit (GPU) Cloud-Native Network Function (CNF) written in NVIDIA cuTile-Rust. The proposed data plane is evaluated against adversarial Large Language Model (LLM) agents on a live six-node Open5GS/UERANSIM testbed. A Red Agent autonomously mutates the implant while a Blue Agent hunts it; the presented behavioral invariant, which requires a functional passive backdoor to hold an `AF_PACKET/ETH_P_ALL/iface=0` socket, is shown to be a kernel fact that the attacker cannot mutate away. A GPU mutation-space sweep evaluates the full space of 2,555,904 signature variants, over which a static signature is evaded in approximately all cases while the invariant is evaded by none. The proposed placement is measured rather than assumed: eBPF/XDP reaches a hard verifier ceiling, whereas the memory-safe CNF reaches parity with hand-written CUDA C at moderate arithmetic intensity and, at higher intensity, outperforms a straightforward hand-written CUDA C implementation, at an edge power envelope of about 66 W. Across six adversarial memory-safety bug classes the cuTile detector reports zero sanitizer errors, and the invariant holds at 25/25 under three distinct attacker LLMs.

M.-A. Kourtis (corresponding author) · G. Xilouris
Institute of Informatics and Telecommunications, NCSR "Demokritos", Athens 15341, Greece
E-mail: akis.kourtis@iit.demokritos.gr

Keywords 5G core security · Network Function Virtualization · eBPF/XDP · Memory-safe GPU computing · Behavioral invariant detection · Adversarial LLM agents

1 Introduction

The transition of fifth-generation (5G) mobile cores towards cloud-native deployments, in which Network Functions (NFs) execute as containerized software over shared Network Function Virtualization Infrastructure (NFVI), has created new requirements in the design and deployment of telco security in terms of persistent, host-resident threats [1,2]. While Network Function Virtualization (NFV) and Software-Defined Networking (SDN) deliver operational agility and cost efficiency, they concurrently expand the attack surface of the mobile core, since a compromised NF can intercept, modify, or exfiltrate the traffic traversing it. Among the NFs defined by 3GPP, the User Plane Function (UPF) is the highest-value target for a persistent implant: it terminates the GPRS Tunnelling Protocol user-plane (GTP-U) tunnels from every base station, decapsulates subscriber traffic, and forwards it to the data network, thus offering an adversary a vantage point over the entire user plane.

The threat is not hypothetical. BPFDoor, an Advanced Persistent Threat (APT) backdoor documented in attacks against telecommunications operators, opens an `AF_PACKET` raw socket guarded by a Berkeley Packet Filter (BPF) and is activated by a hardcoded "magic" value, in order to receive trigger packets without opening any listening port [3–5]. The magic value is a natural target for signature-based detection, but it is equally natural to mutate. Signature-based Network Intrusion Detection Systems (NIDS) such as Suricata, Snort, and Zeek

match byte patterns in packet payloads, and are therefore inherently brittle against a polymorphic implant that alters its wire-level indicators between scans [6].

Two concurrent technology trends sharpen this problem. The first is the maturation of Large Language Model (LLM) tool-use, which enables autonomous mutation agents that adapt implant parameters faster than a human analyst can author a detection signature [14, 15]; the same capability, applied defensively, enables autonomous evidence synthesis from live system interfaces [16]. The second is the arrival of memory-safe GPU programming through NVIDIA cuTile-Rust, which allows a network-facing detection kernel to be authored in safe Rust and compiled natively to the GPU, in order to remove an entire class of memory-safety defects from a security-critical function [27,28]. The convergence of these two stacks, adversarial agentic Artificial Intelligence (AI) and a memory-safe GPU-accelerated data plane, over a live 5G core, is the subject of this work; neither stack alone answers the underlying network service management question of whether a deployable detector exists for adaptive user-plane implants.

Aimed at addressing these concerns concurrently, this work presents a tiered detection data plane and evaluates it under adversarial LLM agents on a fully functional 5G testbed. The study is organized around four research questions: (i) can an LLM Red Agent evade a static signature by mutating the implant; (ii) is there a behavioral invariant the attacker cannot evade, and does an LLM Blue Agent detect it; (iii) does the invariant generalize beyond BPFDoor; and (iv) can the detection function be realized as a memory-safe, GPU-accelerated CNF in the 5G data plane, and where should it be placed. The main contributions of the presented work are the following. A behavioral invariant for passive user-plane implants is identified and shown, through a GPU-parallel sweep of the full mutation space, to be evaded by none of 2,555,904 signature variants, whereas the static signature is evaded in approximately all of them. A tiered data plane is proposed that combines in-kernel eBPF/XDP filtering with a memory-safe cuTile CNF, and its placement crossover is measured rather than assumed. The memory-safety cost of the CNF is quantified against hand-written CUDA C and shown to vanish, and then reverse, as arithmetic intensity grows, while a systematic fault-injection campaign over six bug classes reports zero errors for the safe kernel. Finally, the invariant is evaluated against the tier boundary itself under three distinct attacker LLMs and a closed-loop attacker that adapts to the detection verdict, and is found to hold at 25/25 in every case.

The remainder of the paper is organized as follows. Section 2 discusses the enabling technologies and related work. Section 3 formalizes the threat model and the behavioral invariant. Section 4 develops a Stackelberg model of the interaction and shows that the invariant induces a defender strategy whose evasion set is empty. Section 5 presents the system architecture and the tiered detection data plane. Section 6 reports the experimental evaluation, and Section 7 discusses limitations and deployment. Finally, Section 8 concludes the paper.

2 Background and Related Work

2.1 BPFDoor and passive Linux backdoors

BPFDoor was publicly disclosed in 2021 and subsequently analyzed by several security vendors [3,4]. Its defining characteristic is the use of an `AF_PACKET` raw socket with an attached BPF filter, in order to receive trigger packets without holding any listening port, which evades conventional port scanning entirely. The implant has been documented in attacks against telecommunications providers, and remains a live concern for operators [5]. Several sibling families share the same socket mechanism while differing in how they hide: Symbiote hooks the C library (libc) to subvert user-space enumeration of `/proc`, whereas Drovorub and Reptile employ a kernel module and therefore subvert both `/proc` and `ss`. A common property is retained across the family: the sniffing socket is a kernel fact required for the implant to function, and only its visibility, not its existence, is manipulated by the hiding technique.

2.2 5G core, NFV, and eBPF-based monitoring

The security of the 5G core has been studied predominantly at the control plane and radio layers, including formal verification of authentication, protocol-level vulnerability discovery, and moving-target defense for network slices and for the Packet Forwarding Control Protocol (PFCP) [9,10]. Recent work has also shown that protocol tunneling in 5G can be exploited in order to bridge network boundaries [11], which places the tunneled user plane itself within the attack surface of the mobile core. eBPF and the eXpress Data Path (XDP) have emerged as the dominant in-kernel mechanism for line-rate packet processing and monitoring. Recent work surveys eBPF-based cybersecurity mechanisms [6], examines detector and architecture trade-offs for embedded NIDS built on eBPF/XDP [7], and applies eBPF to monitor 5G core control-plane interfaces [8]. These works operate either at the control plane or at the network level; the presented study is complementary in that

it targets a persistent implant resident within a user-plane NF, and treats the detection function itself as a lifecycle-managed CNF. In contrast to the detector and architecture trade-offs examined within the eBPF/XDP verifier envelope in [7], the placement analysis presented here measures the crossover past the verifier ceiling, at which the fully-unrolled program is rejected, and into a memory-safe GPU function.

2.3 GPU-accelerated packet processing

GPU acceleration of packet processing and deep packet inspection has an established lineage, and inline packet processing frameworks that pair GPUs with the Data Plane Development Kit (DPDK) and GPUDirect are documented enablers for high-throughput analysis [12, 13]. This prior art is, without exception, written in unsafe C or CUDA C. The presented work departs from it by authoring the compute-heavy detection kernel in memory-safe cuTile-Rust and by measuring the memory-safety property as an operational characteristic of the resulting CNF.

2.4 LLM agents for offensive and defensive security

The application of LLMs to offensive security has expanded rapidly, from autonomous penetration testing to agentic red-team frameworks that plan and execute multi-step intrusions [14,15]. On the defensive side, autonomous cyber-defense agents synthesize evidence, triage alerts, and perform forensics over live or collected telemetry [16,19]. Closest in spirit to the presented design is the tiered agentic architecture, in which a deterministic tier handles threats that carry unambiguous indicators while an LLM reasoning tier addresses those that require cross-source correlation [20]. The presented work shares the tiering intuition but differs in target and mechanism: rather than reducing enterprise Security Information and Event Management (SIEM) telemetry for an LLM decision layer, it places an in-kernel eBPF/XDP filter beneath a memory-safe GPU CNF on a 5G UPF, and its two agents form a closed adversarial loop in which the attacker is itself an LLM that mutates a live implant.

2.5 Behavioral and provenance-based detection

Behavioral and provenance-based detection seeks properties that persist across payload mutation, from syscall-sequence profiling to provenance-graph learning for APT detection [17]. Host-based detection of BPFDoor is, in operational practice, performed by enumerating anomalous raw packet sockets through `/proc/net/packet` or `ss`, an approach recommended in the vendor analyses cited above [3–5] and packaged in community detection tooling [18]. The presented work departs from that practice on three concrete grounds: (i) the invariant is formulated as the `ETH_P_ALL/iface=0` binding tied to multi-interface, GTP-U-decapsulated trigger reception; (ii) its un-mutatability is established near-exhaustively rather than asserted; and (iii) the detection function is designed to be hosted in a network-side GPU CNF that observes from outside the compromised host, in order to survive the user-space and kernel hiding that defeats host-resident scanners (Section 6.9). The proposed invariant is thus different in kind from provenance and syscall-sequence learning: it is a single socket-level property, checkable through one kernel-interface query, that is mechanistically required for the implant to function and therefore cannot be mutated away, which is what makes it a deterministic detection primitive rather than a learned classifier.

2.6 Memory safety in systems and network functions

Memory-safety defects remain the dominant class of exploitable vulnerabilities in systems software, which has driven the adoption of Rust in operating-system and networking code, including the ongoing integration of Rust into the Linux kernel [28] and the arrival of memory-safe GPU-kernel programming [27]. The presented CNF extends that trajectory to the accelerator: the compute-heavy detection kernel is authored in memory-safe cuTile-Rust and, distinctively, the memory-safety property is treated as an operational characteristic that is measured rather than merely asserted (Section 6.6).

2.7 Game-theoretic security and Stackelberg models

The leader–follower structure of a committed defense against an adaptive attacker is naturally captured by Stackelberg security games, whose theory and deployments have been surveyed over more than a decade [21], and whose equilibrium relationship to the Nash solution is well characterized [22]. Game-theoretic formulations underpin moving-target defense [24,9], adversarial classification [23], and, most recently, the adversarial co-evolution of malware and detectors cast as a bilevel optimization [25]; Stackelberg models combined with deep reinforcement learning have also been applied to cyber-physical defense [26]. The presented work adopts

the Stackelberg framing but reaches an atypical conclusion: rather than randomizing a resource-constrained defense, the identified invariant yields a defender strategy whose evasion set is structurally empty, so the leader dominates independently of the follower's capability. Section 4 develops this formally.

2.8 Positioning and comparative analysis

Table 1 positions the presented work against representative systems across six dimensions that are relevant to cloud-native 5G detection: (i) a user-plane implant target; (ii) the use of adversarial LLM agents; (iii) a memory-safe detector; (iv) a deterministic rather than learned detection primitive; (v) a measured tiered eBPF/GPU placement; and (vi) an explicit game-theoretic formalization. No prior system combines these six properties. In particular, the closest tiered agentic defender [20] operates on enterprise SIEM telemetry with a learned reasoning tier and no memory-safety or 5G-data-path treatment, whereas the closest game-theoretic malware defense [25] reaches an equilibrium through iterative retraining rather than through a mechanistically un-mutatable invariant.

3 Threat Model and Behavioral Invariant

The adversary is assumed to have achieved initial access to the UPF host and to be able to deploy binaries with root privileges; the goal is persistent, stealthy backdoor access that survives defensive scanning. Three capability tiers of increasing sophistication are considered: (i) magic mutation, in which only the two-byte trigger constant is altered, which suffices to evade a static signature; (ii) protocol and process mutation, which additionally alters the checked Internet Protocol (IP) protocol and masquerades the process name; and (iii) structural mutation, which alters the BPF filter structure, the payload offset, and the socket binding.

The presented behavioral invariant is the following. In order to receive trigger packets across all interfaces, including GTP-U-decapsulated packets arriving on the tunnel interface, a functional passive backdoor must open an `AF_PACKET` raw socket bound to `ETH_P_ALL` (protocol `0x0003`) with interface index zero (`iface=0`). Binding to a specific interface would sacrifice at least one trigger path, and is therefore not available to a functional implant. Legitimate monitoring tools bind to a specific interface, which yields a zero-false-positive discriminator. The interaction between the mutation agent and the detector is a leader–follower game. Its central property, that the invariant leaves the attacker no functional evasion, is formalized in Section 4 and evaluated near-exhaustively in Section 6.

4 A Stackelberg Model of Adaptive Evasion and Invariant Detection

The interaction between the mutation agent and the detector is formalized as a two-player Stackelberg security game [21, 22], in which the defender is the leader that commits to a detection strategy and the attacker is the follower that observes the committed strategy and best-responds. In this respect, the presented model formalizes the central claim of the study: the behavioral invariant induces a defender strategy whose evasion set is empty, so that the leader dominates independently of the follower's capability.

4.1 Players, strategies, and functionality

The defender commits to a strategy $d \in \mathcal{D} = \{d_{\text{sig}}, d_{\text{inv}}\}$, where d_{sig} matches a fixed wire signature and d_{inv} queries the socket table for the invariant. The attacker selects a mutation a from the action space $\mathcal{A}$ induced by the three capability tiers of Section 3; for the swept filter dimensions, $|\mathcal{A}_{\text{filter}}| = 2^{16} \times 13 \times 3 = 2{,}555{,}904$. Let $\phi_{\text{func}}(a) \in \{0, 1\}$ indicate that the mutated implant remains a functional passive backdoor, i.e., that it receives the trigger across all interfaces including the GTP-U-decapsulated path, and let $\phi_{\text{inv}}(a) \in \{0, 1\}$ indicate that the resulting socket satisfies $\texttt{proto} = \texttt{ETH_P_ALL} \wedge \texttt{iface} = 0$.

4.2 Claim A: the invariant is a necessary condition

The central structural fact is the logical necessity

$$\phi_{\text{func}}(a) \;\Rightarrow\; \phi_{\text{inv}}(a), \qquad \forall a \in \mathcal{A}, \tag{1}$$

with contrapositive $\neg\phi_{\text{inv}}(a) \Rightarrow \neg\phi_{\text{func}}(a)$: any binding to a specific interface forfeits at least one trigger path, in particular the tunnel interface on which the decapsulated packets arrive. This direction is measured by the interface-binding ablation of Section 6.10, in which `iface=0` is the unique binding that receives the trigger on both the N6 and the GTP-U path. Writing $\mathcal{A}_{\text{func}} = \{a : \phi_{\text{func}}(a) = 1\}$ for the functional region, (1) states $\mathcal{A}_{\text{func}} \subseteq \{a : \phi_{\text{inv}}(a) = 1\}$.

Table 1 Comparative positioning against representative systems. ✓: supported; P: partial; –: not addressed.

Work	5G/UPF target	LLM agents (red/blue)	Memory-safe detector	Deterministic invariant	Tiered eBPF+GPU	Game-theoretic
MERLINS (MTD/NFV) [9]	✓	–	–	–	–	✓
Provenance IDS (Slot) [17]	–	–	–	–	–	P
Embedded eBPF NIDS [7]	–	–	–	–	P	–
GPU DPI (GASPP) [13]	–	–	–	–	P	–
LLM red-team [15]	–	✓	–	–	–	–
Tiered LLM defender [20]	–	✓	–	P	P	–
Bilevel co-evolution [25]	–	–	–	–	–	✓
This work	✓	✓	✓	✓	✓	✓

4.3 Evasion entropy and the empty evasion set

Let $\mathrm{Det}(d, a) \in \{0, 1\}$ be the detection indicator and let the *evasion set* of a strategy be the functional mutations it fails to detect,

$$E(d) = \{a \in \mathcal{A}_{\text{func}} : \mathrm{Det}(d, a) = 0\}, \tag{2}$$

with evasion entropy $H(d) = \log_2 |E(d)|$ measuring the attacker's freedom. For the signature only the canonical point matches, so $|E(d_{\text{sig}})| = |\mathcal{A}_{\text{func}}| - 1$ and

$$H(d_{\text{sig}}) = \log_2\big(|\mathcal{A}_{\text{func}}| - 1\big) \approx 21.3 \text{ bits} \tag{3}$$

over the swept space, and 16 bits over the magic-only subspace. For the invariant, (1) gives $\mathrm{Det}(d_{\text{inv}}, a) = \phi_{\text{inv}}(a) = 1$ for every $a \in \mathcal{A}_{\text{func}}$, hence

$$E(d_{\text{inv}}) = \varnothing, \qquad H(d_{\text{inv}}) = -\infty, \tag{4}$$

so no functional mutation evades.

4.4 The invariant as a dominant strategy

In a Strong Stackelberg Equilibrium the follower best-responds to the committed d and breaks ties in the leader's favour. Because $E(d_{\text{inv}}) = \varnothing$, every attacker action satisfies either $\neg\phi_{\text{func}}(a)$, which corresponds to a broken and therefore useless implant, or $\mathrm{Det}(d_{\text{inv}}, a) = 1$, which corresponds to detection. With defender utility $u_D = -u_A$ and any attacker payoff that rewards an undetected functional implant, this yields

$$u_D(a, d_{\text{inv}}) \;\geq\; u_D(a, d'), \qquad \forall a \in \mathcal{A},\ \forall d' \in \mathcal{D}, \tag{5}$$

so d_{inv} is a dominant defender strategy and the first-mover disadvantage of a committed defense vanishes. The follower's ability to observe and adapt confers no benefit, since the evasion set is empty as a consequence of (1) rather than of resource allocation. It should be noted that this differs from canonical security games [21,23], in which the defender must randomize a limited resource and the attacker retains a positive evasion probability. The proposed model therefore predicts that the invariant holds independently of the attacker, which is evaluated in Section 6.12 against three distinct attacker LLMs and against a closed-loop attacker that observes the detection verdict and adapts; the invariant holds at 25/25 in both settings, as an empty evasion set predicts.

4.5 Claim B: implementation reliability

Claim A concerns the invariant's presence, which is mechanistic and holds over 100% of $\mathcal{A}_{\text{func}}$. The realized detector $\hat{d}_{\text{inv}}$, however, synthesizes its verdict through an LLM and may fail to report a present invariant, so its per-round reliability $\rho = \Pr[\mathrm{Det} = 1 \mid \phi_{\text{inv}} = 1]$ is below one in implementation. The evaluation estimates $\hat{\rho} = 25/25$ per attacker, with a Wilson 95% interval of $[0.87, 1.00]$ (Section 6.8). The residual $1 - \rho$ is attributed to structured-output error rather than to ϕ_{inv}, since a deterministic `/proc/net/packet` fallback that performs no LLM synthesis is measured at $\hat{\rho}_{\text{fallback}} = 30/30 = 1.00$ (Wilson $[0.886, 1.00]$) across canonical and mutated implants (Section 6.13). The signature strategy, by contrast, cannot be repaired by reliability, since even a perfectly reliable matcher inherits the evasion entropy of (3), and an adaptive signature that adds a rule per observed magic is measured to cover only 2.66×10^{-5} of the combinatorial action space after 100 rounds, while every novel magic evades on first appearance (Section 6.11).

5 System Architecture

The proposed architecture is designed to provide a deployable detection data plane for cloud-native 5G user-plane functions, integrating an in-kernel filtering tier, a memory-safe GPU analysis tier, and an adversarial agentic evaluation loop. The system is structured into complementary layers, in order to ensure line-rate operation, a bounded data-path cost, and a memory-safe

analysis function, and its high-level organization is depicted in Figure 1.

The primary ambition of the architecture is to place each detection workload where it is cheapest, in order to keep the common case in the kernel while reserving the accelerator for compute-heavy analysis. This tier is realized as follows.

The eBPF/XDP filtering tier executes in-kernel on the UPF receive path and performs parsing together with the behavioral-invariant and light-signature checks. It operates at line rate with no user-space copy and no accelerator, and it escalates only suspicious or sampled flows to the analysis tier. This tier is subject to a hard expressiveness ceiling imposed by the in-kernel verifier, which is measured in Section 6.

The memory-safe GPU analysis tier is a Cloud-Native Network Function authored in NVIDIA cuTile-Rust and executed on a low-power inference GPU. It performs the compute-heavy analysis that the verifier cannot express, such as the exhaustive mutation-space sweep and, prospectively, deep inspection or machine-learning classification. The tier is memory-safe by construction, is elastic with a sub-second spin-up, and is multi-tenant under the Multi-Process Service (MPS).

The adversarial agentic loop comprises a Red Agent and a Blue Agent, each a tool-augmented LLM, together with an orchestrator. The Red Agent mutates, recompiles, and redeploys the implant on the UPF, while the Blue Agent queries live kernel interfaces and the tiered detector, in order to synthesize a structured verdict. The orchestrator owns trigger injection and records ground truth from the compiled implant rather than from the agent's claimed action.

The testbed that realizes the architecture is a six-node Proxmox laboratory, summarized in Table 2. The user plane is provided by Open5GS with a UERANSIM base station and User Equipment (UE), and both the direct N6 and the GTP-U-encapsulated trigger paths are exercised. The GPU host carries an NVIDIA L4 (compute capability 8.9), driver 580.178.04, and CUDA 13.2, over which the cuTile toolchain and the local LLM inference runtime execute.

6 Experimental Evaluation

The experimental evaluation was performed on the six-node testbed of Table 2, with the GPU experiments executed on the NVIDIA L4 host. Throughout, timing figures are reported as the mean of at least five repetitions, detection rates are reported with Wilson 95% confidence intervals where a rate is estimated, and the memory-safe GPU kernel is compared against an identically-computing hand-written CUDA C twin. The evaluation separates the security question (Sections 6.1–6.2) from the systems question of placement, cost, and safety (Sections 6.3–6.9), and corroborates the formal claims of Section 4 through the ablation and attacker-variation experiments of Sections 6.10–6.13.

6.1 Agentic Red versus Blue baseline

In the baseline adversarial loop, the Red Agent mutates the implant while the Blue Agent inspects the host invariant together with the Suricata verdict. The behavioral invariant is detected in all 25 of 25 rounds (Wilson 95% confidence interval $[0.87, 1.00]$), while the static Suricata signature is evaded once the compiled magic value departs from its canonical constant. This baseline establishes the two facts the remaining experiments quantify near-exhaustively: the signature is brittle under mutation, whereas the invariant is not.

6.2 Mutation-space sweep

In order to move the evasion-entropy claim from a sampled to a near-exhaustive result, we evaluated the full mutation product space of 2^{16} magic values, 13 payload offsets, and three protocols, i.e., 2,555,904 variants in total, with a cuTile kernel on the L4, together with a corpus-based classic-BPF emulation. The static signature is evaded across essentially the entire space (a single canonical point is detected), which corresponds to about 21.3 bits of signature-evasion entropy, whereas the behavioral invariant is evaded by none of the variants. This is the near-exhaustive form of the Stackelberg claim of Section 4: widening the attacker's signature-evasion freedom leaves the invariant's evasion set empty ($|E(d_{\text{inv}})| = 0$, $H(d_{\text{inv}}) \to -\infty$), so no functional mutation evades.

6.3 Placement crossover and data-path cost

The proposed tiering rests on a measured placement crossover rather than an assumed one. Figure 2 reports the per-packet cost of the same detection at three execution points as a function of the per-packet compute K. The in-kernel eBPF/XDP program is line-rate for light filtering, at 13–75 ns per packet, but the verifier rejects the fully-unrolled program at $K \geq 96$. The Central Processing Unit (CPU) scales linearly with K and has no ceiling, whereas the GPU, for resident data, remains inexpensive and is the only execution point that both fits

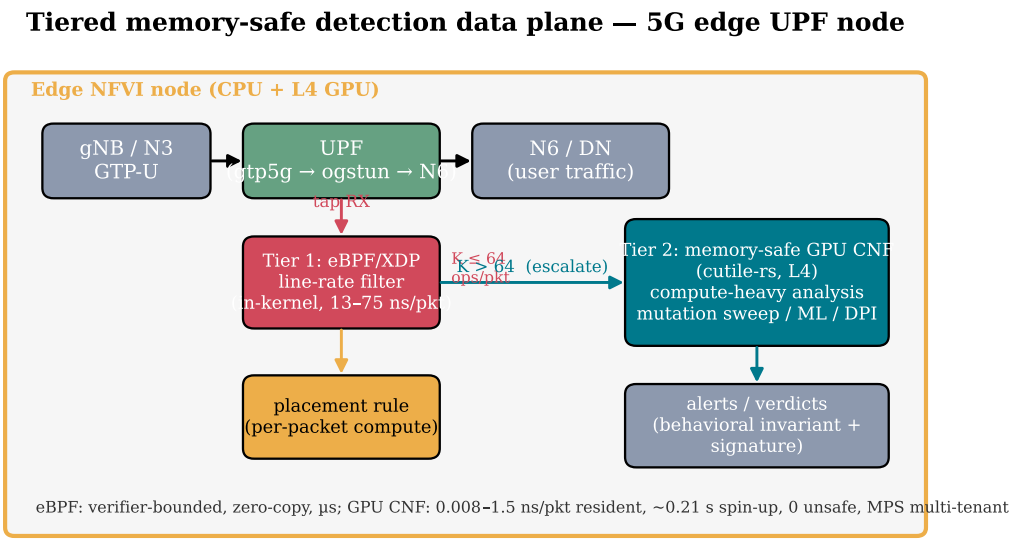


Fig. 1 Tiered, memory-safe detection data plane for a cloud-native 5G UPF: an in-kernel eBPF/XDP filtering tier escalates a pre-filtered subset to a memory-safe cuTile GPU CNF for compute-heavy analysis.

Table 2 Six-node testbed configuration.

Node	Role	Key software
vm1	5G control plane	Open5GS 2.8.0 NFs, MongoDB
vm2	UPF + implant + NIDS	gtp5g kernel module, bpf-listener implant, Suricata 8.0.7
vm3	RAN simulation	UERANSIM v3.2.7 (gNB + UE)
vm4	Red trigger sender	hardened L2 trigger delivery
vm5	Blue agent	remote detection over SSH
vm6	Orchestrator + GPU/LLM	NVIDIA L4 (sm_89), CUDA 13.2, cuTile, Ollama

and scales for heavy per-packet compute, reaching approximately 144× the cost-efficiency of a six-core CPU at $K = 4096$.

At the data-path level, a user-space detection tap on the live N6 interface is measured to cost approximately 45% of throughput and about half of a CPU core under load, which is the concrete motivation for moving light filtering into the kernel and reserving the accelerator for the pre-filtered remainder. It should be noted that the testbed employs virtio interfaces, so the reported figures are indicative of the overhead ratio rather than of an absolute line rate.

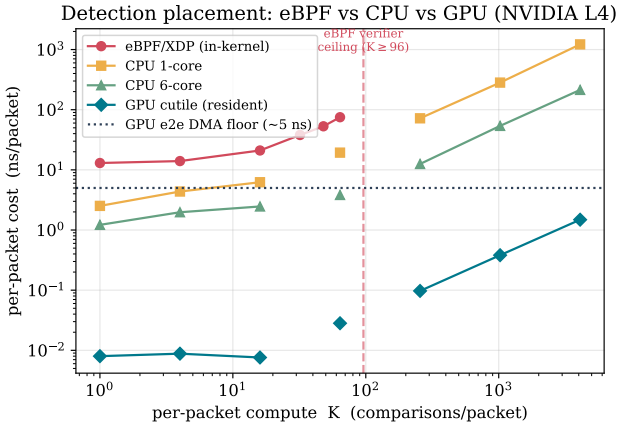


Fig. 2 Placement crossover: per-packet cost (ns) of eBPF/XDP, CPU, and the GPU CNF as a function of per-packet compute K. The eBPF verifier ceiling and the GPU Direct Memory Access (DMA) floor are annotated.

6.4 CNF characterization

The memory-safe detection kernel is characterized as a Cloud-Native Network Function in the terms by which a CNF is judged, namely elastic spin-up, steady-state latency, throughput and scaling, density, and energy. The results are summarized in Table 3 and Figure 3.

Cold-start is dominated by a one-time just-in-time compilation of approximately 0.21–0.23 s and is independent

Table 3 cuTile CNF characterization on the NVIDIA L4 across the mutation space.

Space (mut.)	Warm (μs)	Throughput (mut/s)	VRAM (MB)	Power (W)
9,984	20	1.2×10^9	0.2	30.5
159,744	15	1.8×10^{10}	2.6	38.6
638,976	27	4.9×10^{10}	10.2	49.4
2,555,904	38	9.8×10^{10}	40.9	66.3

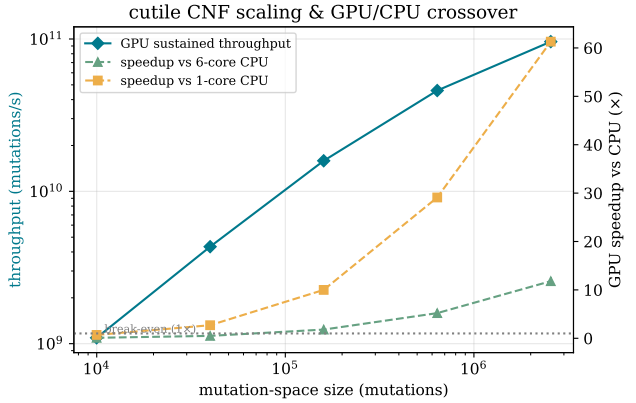


Fig. 3 CNF sustained throughput and speedup over a CPU baseline as a function of the mutation-space size; the GPU CNF overtakes a single core at approximately 40k mutations and a six-core CPU at approximately 160k.

of problem size, so a CNF instance goes from cold to serving in about 0.22 s. Warm steady-state latency for a full sweep is 15–38 μs, which is roughly six orders of magnitude below the 22.7 s of an LLM-in-the-loop detection round. Sustained throughput scales with problem size to approximately 9.8×10^{10} mutations per second at 70% utilization. The per-sweep data footprint is small, of the order of 41 MB, whereas the full per-instance footprint, dominated by the CUDA context rather than by the data, is approximately 224 MB. On the 23 GB L4, this permits on the order of 100 concurrent detection instances, and only under the Multi-Process Service (MPS), since naive co-scheduling collapses the aggregate throughput. The energy cost is approximately 0.69 μWh per full-space sweep at a peak of about 66 W, which keeps the energy efficiency of the CNF within the edge power envelope of the L4, below its 72 W rated board power.

6.5 Memory-safety cost versus arithmetic intensity

A recurring objection to a memory-safe accelerator is a performance tax. In order to quantify it honestly, we added a functionality-preserving compute knob K, realized as K bijective `xorshift` rounds per element, to both the cuTile kernel and the CUDA C twin, and

Table 4 Safety overhead versus arithmetic intensity K (warm latency, mean of five repetitions). The detected count is bit-identical across the twins at every K.

K	cuTile (ms)	CUDA C (ms)	ratio
1	0.073	0.020	3.72×
16	0.072	0.031	2.31×
32	0.074	0.048	1.54×
64	0.082	0.084	0.98×
256	0.202	0.296	0.68×
4096	2.934	4.634	0.63×

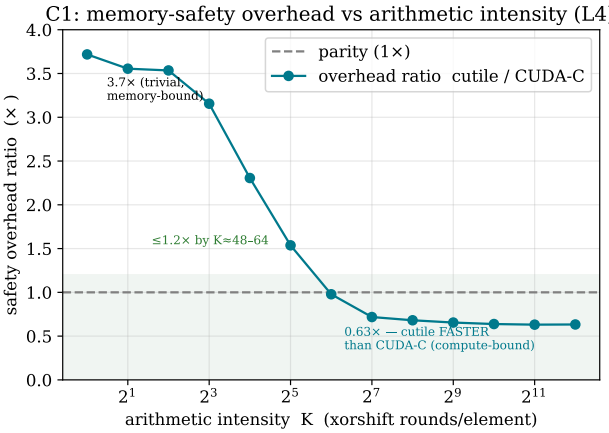


Fig. 4 Memory-safety overhead (cuTile-to-CUDA C warm-latency ratio) as a function of arithmetic intensity K; parity is reached at $K \approx 64$ and the safe kernel is faster beyond it.

held the detected count bit-identical for every K as a correctness gate. Table 4 reports the warm-latency ratio of the two twins as K grows. The overhead is 3.72× at the trivial memory-bound extreme, reaches parity at $K \approx 64$, and inverts to approximately 0.63× at high intensity, where the cuTile compiler optimizes the intensive kernel better than the naïve CUDA C loop. In other words, the frequently-cited threefold safety tax is an artifact of a trivial memory-bound kernel; once per-element compute is realistic, memory safety is effectively free, and beyond that the safe kernel is faster. The full 13-point sweep is depicted in Figure 4.

6.6 Memory-safety fault-injection campaign

The memory-safety property is evaluated systematically rather than anecdotally, through a matrix of six adversarial bug classes against the NVIDIA `compute-sanitizer` tools, as reported in Table 5 and Figure 5. For each class, we exercised a CUDA C twin that intentionally omits the corresponding guard together with the safe cuTile kernel over an adversarial input corpus. Every CUDA C twin triggers its targeted class, which confirms that the tests are live rather than vacuous, while the cuTile ker-

Table 5 Memory-safety fault matrix: CUDA C twin (bug present) versus the safe cuTile kernel, under NVIDIA `compute-sanitizer`.

Bug class	Tool	CUDA C	cuTile
Out-of-bounds read	memcheck	2	0
Out-of-bounds write	memcheck	925	0
Use-after-free	memcheck	925	0
Data race	racecheck	1	0
Uninitialized read	initcheck	924	0
Memory leak	memcheck	1	0

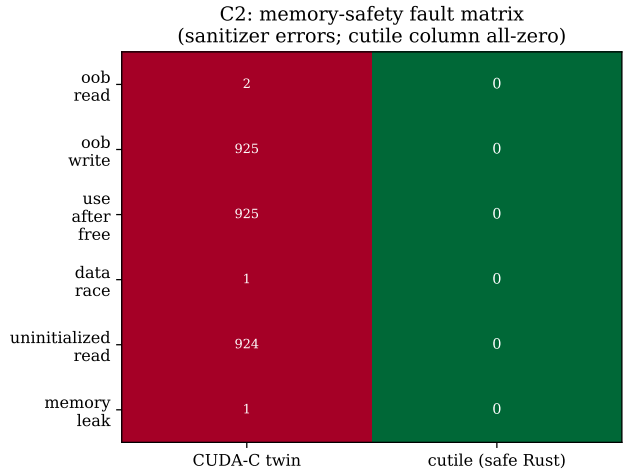


Fig. 5 Memory-safety taxonomy: for each bug class the CUDA C twin triggers the sanitizer while the safe cuTile kernel reports zero.

nel reports zero errors, hazards, and leaks across every cell. The safe tile and tensor Application Programming Interface (API) cannot express the buggy construct: out-of-bounds access requires an `unsafe` raw pointer, use-after-free and leaks are prevented by ownership and Resource Acquisition Is Initialization (RAII), uninitialized reads by typed allocation, and shared-memory races by the disjoint-partition access model. The detection function is thereby shown to eliminate an entire class of defects from a network-facing component.

6.7 Streaming-ingest mitigation

For streaming ingest, the end-to-end throughput of the GPU path is bounded by host-to-device transfer rather than by compute. The presented mitigation combines pinned host memory with multiple streams and double buffering, and its effect is depicted in Figure 6: pinned memory alone yields a 2.34× improvement over the pageable baseline, and streaming adds a further increment to 2.62×, while an approximately tenfold residual to the resident-data ceiling remains and is attributable to the transfer of minimum-size frames. The known remaining

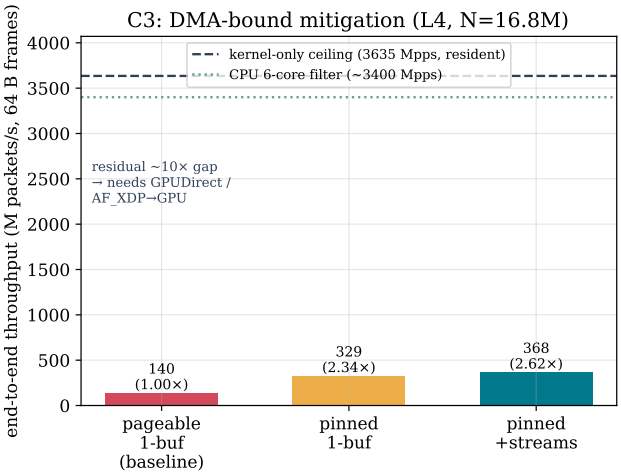


Fig. 6 Streaming-ingest mitigation: end-to-end throughput for pageable, pinned, and pinned-plus-streamed transfers, with the resident-data ceiling annotated.

fix is zero-copy delivery through GPUDirect, which is not available on the present testbed; the software ceiling is therefore reported and line rate is not claimed. This result motivates the tiered design, which escalates only a pre-filtered subset to the accelerator, so that the transfer cost is amortized by heavy per-packet work.

6.8 Red Agent against the tier boundary

The tiered detector is finally evaluated adversarially, with the Red Agent objective changed from evading the implant's signature to driving detection across the tier boundary. Table 6 reports the outcome over 25 rounds for two distinct attacker LLMs against a fixed Blue detector. The Tier-1 signature is evaded to a degree that depends on attacker capability, with 3/25 caught for the stronger attacker and 14/25 for the weaker one, which correspondingly escalates 88% and 44% of rounds to the analysis tier; a signature-only tier is thus only as effective as the attacker is clumsy. By contrast, the Tier-2 behavioral invariant is detected in 25 of 25 rounds under both attackers, which is the central result of this experiment: holding the detector fixed, the invariant is attacker-model-agnostic. The 25/25 invariant rate corresponds to a Wilson 95% lower bound of 0.87, so the property is supported even under the sampling uncertainty of the round count, whereas the Tier-1 signature rates differ sharply across attackers (3/25, Wilson $[0.04, 0.30]$ against the stronger attacker; 14/25, Wilson $[0.37, 0.73]$ against the weaker one). The cross-model comparison is depicted in Figures 7 and 8. We obtained deterministic trigger delivery, required for a fair Tier-1 measurement, by correcting a Proxmox proxy-ARP poisoning of the

Table 6 Red Agent against the tier boundary (25 rounds, Blue = qwen2.5:32b).

Metric	Red=qwen32b	Red=llama8b
Tier-1 signature caught	3/25	14/25
Escalated to Tier-2	22/25	11/25
Tier-2 invariant caught	**25/25**	**25/25**
	[0.87,1.00]	[0.87,1.00]

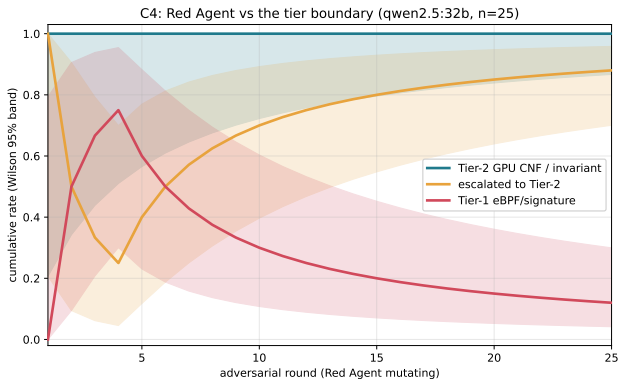


Fig. 7 Cumulative tier detection rates across rounds with Wilson confidence bands for the stronger attacker.

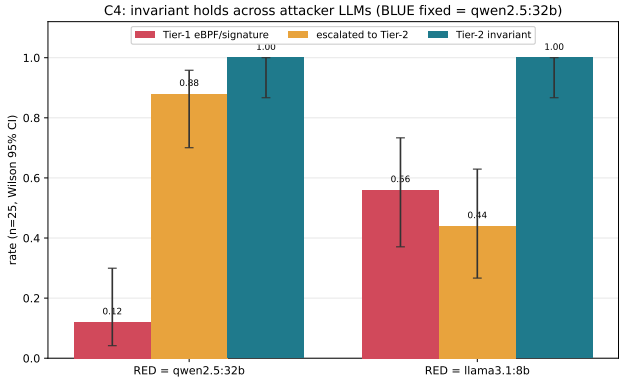


Fig. 8 Cross-model result: the Tier-2 invariant holds at 25/25 under both attacker LLMs, whereas Tier-1 evasion depends on attacker capability.

N6 peer address, which had been the cause of earlier intermittent behavior.

6.9 Generalization beyond BPFDoor

The invariant is finally assessed across sibling families, in order to establish its scope. Because every family considered (BPFDoor, Symbiote, Drovorub, and Reptile) is a passive magic-triggered sniffer that must open a raw socket to function, the invariant is present in all four; what changes is the trusted observation point, which moves up the stack from user-space `/proc` for BPFDoor, to below the hooked libc for Symbiote, to an out-of-kernel or network-side vantage for the kernel-module families. This is a scoping result rather than a claim that one host check catches everything. It constitutes a concrete argument for a network-side GPU CNF that observes from outside the compromised host, since such a vantage remains unaffected by the user-space and kernel hiding that defeats host-based checks for the deeper-hiding families.

Table 7 Interface-binding ablation: triggers received per path (9 sent per cell). `iface=0` is the unique all-path binding.

Binding	N6 (`ens19`)	GTP-U (`ogstun`)	All paths
`iface=0`	9/9	9/9	**yes**
`ens19`	9/9	0/9	no
`ogstun`	0/9	9/9	no
`eth0`	0/9	0/9	no
`lo`	0/9	0/9	no

6.10 Necessity of the invariant: interface-binding ablation

The Stackelberg argument of Section 4 rests on the necessity of (1), whose contrapositive states that a socket bound to a specific interface forfeits at least one trigger path. Whereas the mutation sweep of Section 6.2 varies the filter, this experiment varies the *socket binding* in order to measure the necessity direction directly. We swept the implant's `AF_PACKET` binding over {`iface=0`, `ens19`, `ogstun`, `eth0`, `lo`} and, for each binding, fired the trigger over two interface-distinct paths, namely direct N6 (Ethernet on `ens19`) and GTP-U (raw-IP on `ogstun`), with nine triggers per cell and ground truth read from the running implant. The outcome is reported in Table 7 and Figure 9: the all-interface binding `iface=0` is the unique configuration functional on both paths, while every specific-interface binding forfeits at least one path. This is the measured contrapositive $\neg\phi_{\text{inv}} \Rightarrow \neg\phi_{\text{func}}$ over the interface set, and it grounds Claim A empirically rather than by assertion.

6.11 Adaptive-signature coverage decay

The evasion-entropy gap of (3) is complemented by measuring how a signature defense fares when it is allowed to adapt. We ran a defender that appends one live Suricata rule per observed magic against an attacker that mutates the magic each round, over 100 rounds. The result, in Figure 10, quantifies the asymmetry: every novel magic evades on first appearance (0/67), reused magics for which a rule already exists are caught (33/33, so the rule path is sound), and after 100 rounds the

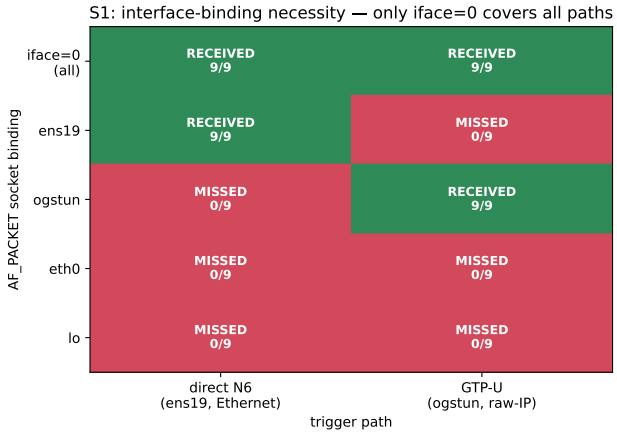


Fig. 9 Interface-binding necessity ablation: only `iface=0` receives the trigger on both the N6 and the GTP-U path, the measured contrapositive of the invariant's necessity.

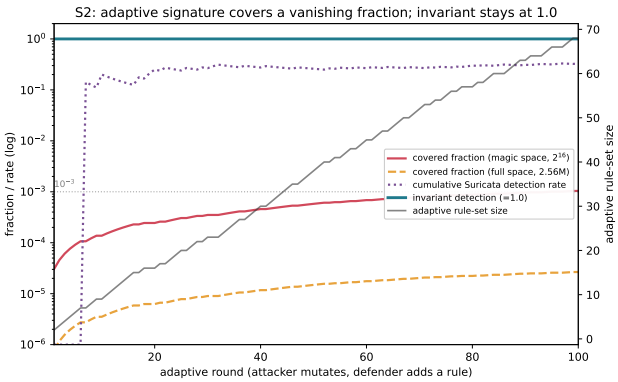


Fig. 10 Adaptive-signature coverage decay over 100 rounds: the covered fraction of the action space stays near 10^{-5} while every novel magic evades on first use; the invariant holds throughout.

rule set covers only 2.66×10^{-5} of the 2,555,904-point action space, and 1.04×10^{-3} of the 2^{16} magic subspace. Throughout, the behavioral invariant is detected in all 100 rounds (Wilson 95% interval $[0.963, 1.00]$), since the mutation moves the magic that the signature chases and never the `iface=0` socket that the invariant keys on. The adaptive signature thus remains structurally one step behind the attacker, which is the operational meaning of a 21.3-bit evasion entropy.

6.12 Attacker-agnosticism and closed-loop co-evolution

The dominance property of (5) predicts that the invariant holds independently of the attacker's capability and even when the attacker adapts to the detector. Two extensions of the tier-boundary experiment of Section 6.8 evaluate this. First, a third attacker model is added, so that the invariant is exercised against qwen2.5:32b, llama3.1:8b, and qwen2.5:14b; as reported in Table 8, the invariant is detected in 25/25 rounds for every model, and only the Tier-1 signature-evasion rate varies with attacker capability. Second, a closed-loop attacker is given the previous round's detection verdict and adapts accordingly; the invariant still holds at 25/25 under adaptation, identical to the open-loop control, which is the empirical signature of an empty evasion set. It should be noted that the closed-loop feedback did not significantly change the Tier-1 signature catch rate (open 7/25 versus closed 10/25, with overlapping intervals), a negative result that is reported as measured. The outcome that matters for (5) is that the invariant is unaffected by adaptation, as depicted in Figure 11. The intended reasoning-model attacker (deepseek-r1:32b) could not serve as a tool-calling agent and was replaced by qwen2.5:14b, which is noted as a limitation.

Table 8 Invariant detection across three attacker LLMs (25 rounds each, Blue = qwen2.5:32b).

Attacker	Tier-1 signature	Tier-2 invariant
qwen2.5:32b	3/25	**25/25**
llama3.1:8b	14/25	**25/25**
qwen2.5:14b	2/25	**25/25**

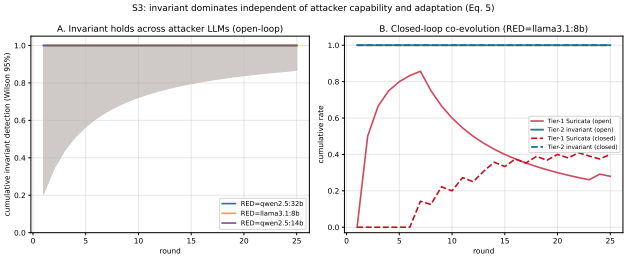


Fig. 11 Attacker-agnosticism: the invariant holds at 25/25 across three attacker models (left) and under closed-loop detection-feedback adaptation (right), whereas signature evasion depends on the attacker.

6.13 Deterministic detection fallback

Claim B attributes the residual $1 - \rho$ of the realized detector to LLM structured-output synthesis rather than to the invariant. A deterministic detector that parses `/proc/net/packet` for `proto` $=$ `0x0003` $\wedge$ `iface` $= 0$ with no LLM in the loop is evaluated over 30 rounds spanning canonical and mutated implants; it detects the invariant in 30/30 rounds ($\hat{\rho}_{\text{fallback}} = 1.00$, Wilson $[0.886, 1.00]$), across sixteen distinct magic values. The invariant's detection by the deterministic path is therefore complete on this sample; the LLM Blue detector was also complete (30/30) on the same rounds, so no synthesis miss occurred here. Since the deterministic

path performs no LLM synthesis, its reliability cannot be limited by structured-output error, which localizes the residual $1 - \rho$ of the realized detector to that step and makes it removable, whereas the signature strategy has no analogous repair.

6.14 Trade-off discussion

The experiments highlight a practical trade-off between the safety and portability of the detection function and its raw latency. The memory-safe CNF accepts a one-time compilation cost of approximately 0.22 s and, on trivial kernels, a warm-latency penalty, in exchange for the elimination of an entire class of memory-safety defects and for a placement that fits within the data-path budget; for the compute-bound analysis that motivates the accelerator, that penalty vanishes and reverses. For streaming ingest the accelerator is not the right tool without zero-copy hardware, which is why the presented design keeps line-rate filtering in the kernel and escalates only the compute-heavy remainder.

7 Discussion and Limitations

The presented evaluation is subject to several limitations that also delineate future work. The measurements were obtained on a single low-power inference GPU and on virtio interfaces, so the data-path figures are indicative of overhead ratios rather than of absolute line rate, and the streaming-ingest ceiling would be lifted by GPUDirect hardware that the testbed does not provide. The generalization result is a static, mechanism-level argument grounded in the mutation sweep and in public reporting, and was not obtained through first-hand execution of the sibling implants. A further scoping concern is the observation vantage. The invariant-detection results were obtained through host-resident kernel queries on the UPF, whereas the threat model grants the adversary root privileges, and a root implant of the libc-hooking or kernel-module class discussed in Section 6.9 can in principle tamper with that query path. The resilience established here should therefore be read as resilience against *socket mutation* by the adversary, not as tamper-proofing of the detection path itself; the network-side, mirror-fed GPU CNF, which observes from outside the compromised host, is the presented response to host-level hiding and tampering, and its full evaluation against an actively tampering implant is left as future work. Finally, the scope of the memory-safety guarantee should be stated plainly: the detection function is memory-safe by construction, whereas the GPU driver stack beneath it is written in C and is not, so the trusted base of the CNF is not yet end-to-end memory-safe. Closing that gap would require a memory-safe kernel driver and is left as future work.

8 Conclusions

This paper presented a tiered, memory-safe detection data plane for cloud-native 5G user-plane functions, which aims to evolve and improve the resilience of intrusion detection against adaptive, host-resident implants. The manuscript not only presented a behavioral invariant that a functional passive backdoor cannot mutate away, but also evaluated its detection under adversarial LLM agents on a live 5G core and characterized the memory-safe GPU function that checks it. Across a near-exhaustive sweep of about 2.56 million signature variants the static signature was evaded in approximately all cases, while the invariant was evaded by none. The placement of the detection function was measured rather than assumed, with the memory-safe CNF reaching parity with hand-written CUDA C at moderate arithmetic intensity and, at higher intensity, outperforming a straightforward hand-written CUDA C baseline, at an edge power envelope of about 66 W. Across six adversarial bug classes the safe kernel reported zero sanitizer errors, while the invariant held at 25/25 under three distinct attacker LLMs and under a closed-loop attacker that adapts to the detection verdict. In contrast to signature-based detection, whose effectiveness was shown to depend on how clumsy the attacker is, the proposed invariant is attacker-model-agnostic. Future work will focus on lifting the streaming-ingest ceiling with GPUDirect delivery, on extending the analysis tier towards a memory-safe kernel-driver trusted base, and on evaluating the design at datacenter scale.

Acknowledgements The research leading to these results has been supported by the PQ-NEXT project, funded by the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101225759, and by the SHARC project, funded by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101290994. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the SNS JU; neither the European Union nor the granting authority can be held responsible for them. The authors thank the NCSR "Demokritos" infrastructure team for testbed support.

Statements and Declarations

Funding. This work was funded by the European Union's Horizon Europe research and innovation

programme through the PQ-NEXT project (Grant Agreement No. 101225759) and, through the Smart Networks and Services Joint Undertaking (SNS JU), the SHARC project (Grant Agreement No. 101290994).

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

Author contributions. M.-A. Kourtis conceived the study, designed and implemented the detection data plane and the adversarial agent loop, performed the experiments and the analysis, and wrote the manuscript. G. Xilouris supervised the work, provided the testbed infrastructure, and reviewed and edited the manuscript. Both authors read and approved the final manuscript.

Ethics approval and consent. Not applicable. The study involves no human participants, personal data, or animals. All experiments were conducted on an isolated laboratory testbed with a laboratory implant; no operational network or third-party system was accessed.

Data and code availability. The orchestrator scripts, the cuTile and CUDA C kernels, the mutation-sweep code, and the raw experimental data supporting the findings of this study are available from the corresponding author on reasonable request, and are intended for release as a public artifact.

References

1. 3GPP, System architecture for the 5G System (5GS), TS 23.501, 3rd Generation Partnership Project, 2023.
2. ETSI, Network Functions Virtualisation (NFV); NFV Security, ETSI GS NFV-SEC series, 2022.
3. Trend Micro Research, BPFDoor's hidden controller used against targets in Asia and the Middle East, Technical Report, 2025.
4. Rapid7, BPFDoor in telecom networks: sleeper cells in the backbone, Threat Research Report, 2025.
5. Ericsson, Minimizing BPFDoor risks for telecom operators, Ericsson Blog, 2025.
6. S. Kostopoulos, P. Tsakonas, and E. K. Markakis, eBPF-based cybersecurity mechanisms: a systematic literature review, arXiv:2608.27511, 2026.
7. S. Wu, O. Koshovyi, G. Pseiridis Pseiras, V. Morel, and R. Duvignau, A feature-rich embedded NIDS with eBPF/XDP: detector and architecture trade-offs, arXiv:2609.12605, 2026.
8. IMDEA Networks Institute, Monitoring 5G core network vulnerabilities with eBPF, Technical Report, 2025.
9. K. Soussi et al., MERLINS: moving-target defense for NFV infrastructure, IEEE NFV-SDN, 2023.
10. A. Landa-Arrue, J. Astorga, I. Garitano, and A. Urbieta, Mitigation of PFCP attacks in 5G networks: dynamic defense through moving target defense and honeynets, in Availability, Reliability and Security (ARES 2025), Lecture Notes in Computer Science, Springer, pp. 141–162, 2025, doi:10.1007/978-3-032-00624-0_7.
11. A. Shaik, R. Jaschek, and J.-P. Seifert, Uncovering hidden paths in 5G: exploiting protocol tunneling and network boundary bridging, in Proc. ACM SIGSAC Conf. on Computer and Communications Security (CCS), pp. 231–245, 2025, doi:10.1145/3719027.3765206.
12. NVIDIA, Optimizing inline packet processing using DPDK and GPUdev with GPUs, NVIDIA Technical Blog, 2023.
13. G. Vasiliadis, L. Koromilas, M. Polychronakis, and S. Ioannidis, GASPP: a GPU-accelerated stateful packet processing framework, in Proc. USENIX Annual Technical Conference (ATC), 2014.
14. M. Mamun, M. Gaber, S. Buffett, and S. Saad, Autonomous adversary: red-teaming in the age of LLM, arXiv:2605.06486, 2026.
15. B. Challita and P. Parrend, RedTeamLLM: an agentic AI framework for offensive security, arXiv:2505.06913, 2025.
16. S. Fumero, K. Huang, M. Boffa, D. Giordano, M. Mellia, Z. Ben Houidi, and D. Rossi, CyberSleuth: autonomous blue-team LLM agent for web attack forensics, arXiv:2508.20643, 2025.
17. W. Qiao, Y. Feng, T. Li, Z. Ma, Y. Shen, J. Ma, and Y. Liu, Slot: provenance-driven APT detection through graph reinforcement learning, in Proc. ACM SIGSAC Conf. on Computer and Communications Security (CCS), 2025.
18. Help Net Security, Researchers release a tool to detect stealthy BPFDoor implants in critical-infrastructure networks, 2026.
19. S. R. Castro, R. Campbell, N. Lau, O. Villalobos, J. Duan, and A. A. Cardenas, Large language models are autonomous cyber defenders, in Proc. IEEE Conf. on Artificial Intelligence (CAI), 2025.
20. S. Boboila, X. Cadet, E. Koh, D. Balasubramanian, D. Van Bruggen, P. Chin, and A. Oprea, BlueSTAR: tiered agentic architecture for autonomous cyber defense, arXiv:2609.11852, 2026.
21. A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe, Stackelberg security games: looking beyond a decade of success, in Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), 2018.
22. D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe, Stackelberg vs. Nash in security games: an extended investigation of interchangeability, equivalence, and uniqueness, Journal of Artificial Intelligence Research, vol. 41, pp. 297–327, 2011.
23. L. Dritsoula, P. Loiseau, and J. Musacchio, A game-theoretic analysis of adversarial classification, IEEE Transactions on Information Forensics and Security, vol. 12, no. 12, pp. 3094–3109, 2017.
24. S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, A survey of moving target defenses for network security, IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1909–1941, 2020.
25. O. Jurečková, M. Jureček, M. Kozák, and R. Lórencz, Adversarial co-evolution of malware and detection models: a bilevel optimization perspective, arXiv:2604.22569, 2026.
26. A. Raptis, S. Gritzalis, and A. N. Yannacopoulos, Dynamic defense strategies for cyber-physical systems using Stackelberg games and deep reinforcement learning in discrete and continuous time, International Journal of Information Security, 2026.
27. NVIDIA, Introducing CUDA Rust: two tracks for writing GPU kernels, NVIDIA Technical Blog, 2026.
28. Internet Security Research Group (Prossimo), An update on memory safety in the Linux kernel, 2025.