Towards Bridging the Telemetry Gap for Security Applications in 6G OpenRANs via eBPF

Haohuang Wen[†§], Vinod Yegneswaran[‡§], Phillip Porras[‡§], Ashish Gehani[‡§], Prakhar Sharma[‡§], Zhiqiang Lin[†§]
[†]The Ohio State University, [‡]SRI, [§]SE-RAN.ai

***Abstract*—The current mobile network is migrating towards a programmable, interoperable, and cloud-native architecture, known as OpenRAN. This enables software-defined services to be integrated as modular applications (xApps and rApps) in a centralized RAN Intelligent Controller (RIC). While prior research has demonstrated a few xApps on OpenRAN for security, optimization, etc., a critical development challenge remains. We observe that a fundamental obstacle is the *Telemetry Gap*: an OpenRAN application has to acquire the necessary analytic telemetry which may not be supported by the corresponding RAN vendors. Unfortunately, the OpenRAN standard does not specify how to address this challenge, and current solutions are typically vendor lock-in, significantly limiting their portability. To bridge this gap, we present our preliminary work on TELERAN, a fully vendor-agnostic agent that enables protocol-level fine-grained visibility and seamless O-RAN integration for virtual RAN nodes at the edge by utilizing extended Berkeley Packet Filter (eBPF). It is driven by two synergistic cross-layer components: (1) an eBPF-based programmable filter that brings in universal and efficient cellular packet filtering at the OS kernel level, and (2) a user-space parser that reconstructs packet semantics based on ASN.1 specifications, enabling operators to customize and program various RAN telemetry. We have implemented a prototype of TELERAN, demonstrating that its seamless integration to two leading open-sourced RAN implementations, OpenAirInterface and srsRAN, with zero source code modification. We also show that TELERAN can be programmed for a wide range of telemetry types for both performance and security analytics, further supporting diverse xApp use cases on OpenRAN.**

I. INTRODUCTION

Cellular networks are the backbone of modern wireless communication, playing an integral part in numerous applications from transportation and entertainment to manufacturing and healthcare. In recent years, the next-generation cellular network is moving towards a software-defined and interoperable architecture, which is considered a major breakthrough over the legacy cellular infrastructures. Such an innovative architecture, also known as Open Radio Access Network or OpenRAN [6]. It disaggregated the monolithic base station into Centralized Units (O-CU) and Distributed Units (O-DU), and decouples the network control logic into a centralized network controller called the RAN Intelligent Controller (RIC) that can be enriched with programmer-defined "plug-n-play" xApps and rApps [3]. This further brings tremendous flexibility and programmability to mobile networks, improving it from various aspects including intelligence, resilience, and security.

However, there exists a critical telemetry gap between OpenRAN application developers and Radio Access Network (RAN) vendors, which hinders OpenRAN adoption in the industry. Specifically, the *telemetry gap* refers to the challenge for the OpenRAN apps to obtain the required telemetry to perform the analysis, which may not match what the connected RAN actually offers. Unfortunately, the OpenRAN standard, specifically the E2 service models (E2SMs) [3] only specifies how telemetry should be reported through the E2 interfaces (via E2SM-KPM [4]), while the telemetry collection is solely implemented by the RAN software vendors. To fill this gap, the mobile network community has proposed solutions for realizing programmable and dynamic telemetry for OpenRAN networks [14, 16, 21, 23, 28, 32]. However, we observe that these solutions fall short as they either require intrusive modifications to the RAN implementations or rely on vendor-specific interfaces to code telemetry generation logic. These drawbacks limit their adoption in complicated 5G deployment in practice, where 6G system integrators may employ multi-vendor RAN solutions, which do not allow modification, recompilation, or reboot during operation.

To bridge this gap, we present TELERAN, the first vendor-agnostic OpenRAN data plane agent, which can generate and report fine-grained and programmable data telemetry to the OpenRAN control plane. The key idea of TELERAN is to apply introspection, a common cybersecurity technique for monitoring the run-time status of virtual machines by using an external program. We migrate this concept to the cellular network domain, enabling it to reconstruct and generate dynamic telemetry of a 5G or 6G RAN, by leveraging its exposed information from standard cellular network interfaces available to the host. The design of TELERAN involves non-trivial technical challenges: (1) how do we efficiently extract desired telemetry from large-volume network traffic? (2) how do we reconstruct cellular packet semantics from raw traffic? (3) how do we design mechanisms to generate vendor-agnostic and standard-compliant cellular telemetry?

Workshop on Security and Privacy of Next-Generation Networks (FutureG) 2026
23 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-9-7
https://dx.doi.org/10.14722/futureg.2026.23040
www.ndss-symposium.org

Contributions. TELERAN addresses these technical challenges by employing novel designs, including a kernel-based efficient packet filter using the Extended Berkeley Packet Filter (eBPF) [15], and a user-space parser that utilizes only 3GPP's ASN.1 specifications as guidance to reconstruct packet semantics for telemetry extraction [10, 12, 13]. These enable several critical benefits. First, it makes TELERAN non-intrusive, indicating that it operates independently outside the native RAN and does not require any modification and recompilation of the RAN software. Second, TELERAN is vendor-agnostic as it only requires standard-level knowledge as guidance, which makes it fully decoupled from proprietary vendor implementations compared to similar frameworks like Janus [16]. This makes TELERAN a "one-size-fits-all" solution that works on any 3GPP-compliant network vendor. Furthermore, the eBPF-based filter ensures efficient, low-latency packet processing at the kernel level, minimizing the impact on performance-critical RAN nodes. By leveraging the eBPF verifier to guarantee kernel stability and prevent system crashes, TELERAN maintains security and broad compatibility across general-purpose Linux servers.

We conducted a lightweight evaluation using a simulated cellular testbed built on OpenAirInterface (v2.1.0) and OpenRAN SC RIC (I-Release). Our results demonstrate that TELERAN introduces minimal system overhead to the RAN, maintaining a $<$1ms control-plane packet processing latency, with peak resource consumption under 4% of a single i5-13500 CPU core and a memory footprint of 11MB. Looking ahead, we expect TELERAN to provide a fully vendor-agnostic and extensible solution to bridge the gap between OpenRAN app and Open CU, DU (OCUDU) [17] developers, further enabling numerous opportunities for innovative OpenRAN applications.

II. MOTIVATION

The practical adoption of OpenRAN faces a fundamental challenge: the *telemetry gap* between the application developers and RAN providers. The fragmented RAN implementations introduce poor extensibility and portability, as they typically extract telemetry data by directly accessing vendor-specific internal data structures and proprietary APIs [7]. Consequently, this fragmentation and the resulting development overhead hinder the interoperability promised by OpenRAN. In the following, we discuss the alternative and existing solutions that could be utilized to address this issue.

Generic Network Packet Parser. There are user-space packet dissectors and parsing tools such as *libpcap* [30], *tcpdump* [19], and *tshark* (or *dumpcap*) [36], some of which are capable of dissecting ASN.1 encoded cellular packets. While these can be alternative solutions, we argue that TELERAN's design has two unique advantages. First, the aforementioned parsing tools incur high performance overhead due to frequent context switches, which impacts the real-time data transmission for the 6G RAN. For instance, our experiment shows that running `dumpcap` to monitor the F1AP interface brought significant latency overhead and caused a UE to disconnect. To mitigate performance issues, TELERAN's

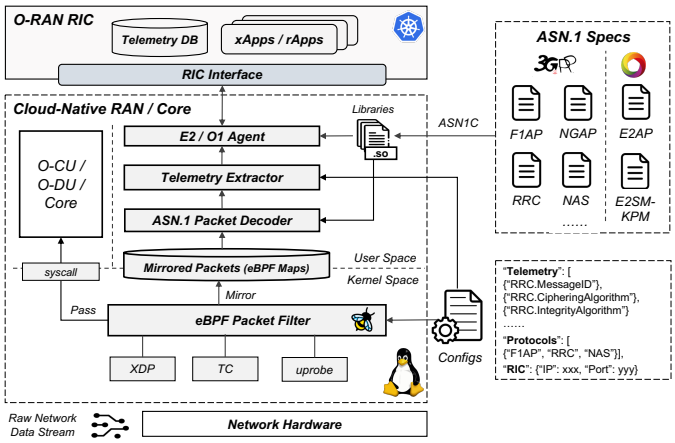

Figure 1: A detailed design view showing TELERAN's internal architecture and workflow.

eBPF-as-a-filter approach significantly reduces overhead as packets are filtered and mirrored at the OS kernel level without affecting normal traffic flow.

Open-sourced RAN Agents. The FlexRIC agent [28] and ONOS E2 agent [1] are two open-sourced implementations dedicated to certain RAN and RIC frameworks. They support several E2 service models and xApps such as E2SM-KPM [4] and RAN Control (RC) xApps [5]. Unfortunately, both agents require extensive instrumentation in the RAN source code, introducing deep dependence on vendor-specific APIs and data structures. Moreover, these agents typically support only a small set of telemetry that does not fulfill the requirement for most RIC xApps, such as those recently developed in the security context that typically require fine-grained information [20, 27, 29, 33].

Janus. As a closely-related framework, Janus [16] also supports dynamic RAN telemetry. Its key components are known as the Janus codelets, which are eBPF-like user-space programs implementing custom telemetry collection and control logic. From its design, we notice that Janus's programmability is not fully decoupled from the vendor implementations, as it relies on hooks that are attached to key interfaces and locations within the RAN code stack. As in practice, RAN software is extremely complex and fragmented [7, 18], which makes it costly to adapt Janus to various vendors. Janus's design indicates its users have to program on top of vendor-defined functions and parameters. Its user-space instrumentation may also incur performance overhead for latency-sensitive RAN nodes.

III. SYSTEM DESIGN

To overcome the limitation with legacy static instrumentation, we introduce TELERAN, a vendor-agnostic agent designed to operate alongside black-box, cloud-native RAN and Core workloads (e.g., O-DUs and O-CUs). As illustrated in Figure 1, TELERAN employs a synergistic cross-layer architecture: it combines an efficient OS kernel-level eBPF

filter to identify relevant traffic with a user-space parser that reconstructs protocol semantics, enabling the programming of customizable telemetry. TELERAN relies solely on a standard Linux environment and 3GPP compliance. By leveraging eBPF [15], it injects dynamic, programmable probes into the kernel's high-performance network datapath without requiring proprietary source code access, binary modifications, or system reboots. This architecture further integrates an independent agent to bridge arbitrary RAN implementations with OpenRAN xApps and rApps, effectively democratizing security analytics across heterogeneous infrastructure. The specific design of these core components is detailed below.

eBPF-Based Cellular Packet Filters. The restricted execution environment of the Linux kernel layer precludes the decoding of complex cellular packets encoded with ASN.1 syntax. Consequently, TELERAN employs lightweight eBPF programs to efficiently filter RAN packets directly at the OS kernel level and offload them to user space, which only incurs <1ms of control-plane packet processing latency for the RAN according to our evaluation within a 5G SA testbed. These eBPF programs instrument critical network data paths to parse raw packet structures and mirror selected packets for cellular-specific processing in user space. Specifically, we leverage the *eXpress Data Path (XDP)*, a high-performance programmable data path available in Linux kernels since version 4.8. XDP enables the filter to intercept and process traffic on critical 5G interfaces such as F1AP and NGAP that contain essential data for RAN and device analytics [10, 11].

As illustrated in Figure 2a, the eBPF filter logic operates by inspecting packet headers. In this example, we utilize standard Linux kernel libraries and helper APIs to recognize SCTP packets to identify F1AP packets that carry UE to base station traffic. Similarly, general-purpose protocols, including Ethernet, UDP, TCP, and IP, are recognized and parsed at the kernel layer, allowing for the effective extraction of encapsulated content destined for higher layers. Once a target RAN packet is identified, it must be forwarded to user space for analysis, as the kernel lacks the necessary libraries for complex ASN.1 decoding. To facilitate this, we employ *eBPF maps*, a shared memory mechanism that bridges kernel and user space. The kernel-side filter stores the relevant RAN packet payloads into these maps. User-space applications subsequently retrieve the data using standard library functions, such as `bpf_map_lookup_elem` in `libbpf`.

Specification-Guided Semantics Reconstruction. A significant challenge in our design is the *semantic gap*, as filtered network packets consist of raw byte sequences without structural context. To bridge this gap, our framework must reconstruct the packet semantics, specifically the protocol structures and field values. We employ a vendor-agnostic approach to achieve this reconstruction in user space. Our methodology leverages the standardized RAN interfaces defined by 3GPP specifications, which describe packet structures using ASN.1 [10, 11, 13]. These definitions can be automatically compiled into source code using off-the-shelf tools such as the ASN1C translator [31]. We utilize these generated artifacts to build modular plugins that facilitate the precise parsing of RAN interface packets. Crucially, this approach decouples the parsing logic from proprietary vendor implementations, ensuring that semantic reconstruction relies strictly on authoritative standards.

```c
SEC("xdp") // Define the eBPF XDP program
int ebpf_filter(struct xdp_md *ctx) {
    void *data = (void *)(long)ctx->data; // Get packet data
    void *data_end = (void *)(long)ctx->data_end;
    ...
    // Check if the IP packet is an SCTP packet
    if (iph->protocol == IPPROTO_SCTP) {
        struct sctphdr *sctp = (struct sctphdr *)(iph + 1);
        if ((void *)sctp + sizeof(*sctp) > data_end)
            return XDP_PASS; // Boundary check
        ...
        // Check if the packet is F1AP over SCTP
        if (proto_id == SCTP_F1AP_ID)
            // Mirror packet to eBPF maps
            bpf_map_update_elem(&map, &key, &packet, BPF_ANY)
    }
    return XDP_PASS; // Pass the packet to user space
}
```

(a) A simplified kernel-level eBPF filter to identify and offload F1AP packets to user space parsers via eBPF maps.

```c
void process_rrc_ul_dcch(UL_DCCH_Message_t *ul_dcch_msg) {
    // Parsing RRC Setup Complete Message
    if (ul_dcch_msg->message.choice.c1->present ==
     UL_DCCH_MessageType__c1_PR_rrcSetupComplete) {
        // Updating KPM counter: RRC Successful Conn
        update_telemetry_cntr("RRC.ConnEstabSucc", 1);
        // Extract the UE TMSI from message
        uint64_t tmsi = str2int(&ul_dcch_msg.ng_5G_S_TMSI)
        update_telemetry("s_tmsi", ueid, tmsi); }
}
```

(b) A simplified user-space telemetry extractor code snippet programmed on top of ASN.1-decoded data structures.

Figure 2: Example filter and telemetry extraction code of TELERAN, working universally across vendors.

Protocol-Aware Telemetry Extraction. Building upon the reconstructed semantic context, TELERAN instantiates specialized handlers tailored to specific protocol families. As demonstrated in Figure 2b (which depicts a handler for RRC Uplink DCCH messages), telemetry extraction logic is embedded directly within these handlers and executes on a per-packet basis. To streamline data aggregation, the framework exposes a suite of helper APIs. The `update_telemetry_cntr` function is designed to manage global counters, facilitating efficient Key Performance Measurement (KPM) collection. Conversely, `update_telemetry` captures granular, packet-specific attributes, such as recording the Temporary Mobile Subscriber Identity (TMSI) for individual User Equipment (UE) tracking. For the final stage of telemetry reporting, TELERAN incorporates an autonomous E2/O1 agent. This agent aggregates the extracted metrics and transmits them to the OpenRAN RIC or Service Management and Orchestration (SMO) [2] that hosts xApps and rApps via OpenRAN compliant interfaces such as E2AP [3] and E2SM-KPM [4]. Strict adherence to these standards ensures seamless interoperability with any OpenRAN RIC platforms.

Telemetry	Category	Handler	LoC
UE C-RNTI	Metadata	F1AP	37
UE S-TMSI	Metadata	RRC	45
UE Cipher Algorithm	Metadata	RRC	35
UE Integrity Algorithm	Metadata	RRC	35
UE RRC State	State	RRC	169
UE NAS State	State	NAS	89
UE RRC Security State	State	RRC	169
UE NAS Security State	State	NAS	89
RRC Message ID	State	RRC	40
NAS Message ID	State	NAS	41

Table I: Security telemetry reproduced in TELERAN and the lines of C code taken to implement them.

KPM	Type	Handler	LoC	Source
RRC.ConnEstabAtt	CC	RRC	16	5.1.1.15.1
RRC.ConnEstabSucc	CC	RRC	16	5.1.1.15.2
RRC.ConnEstabFailCause	CC	RRC	16	5.1.1.15.3
RRC.RelWithoutSuspendConfig	CC	RRC	16	5.1.1.15.4
UECNTX.ConnEstabAtt	CC	NAS	16	5.1.1.16.1
UECNTX.ConnEstabSucc	CC	NAS	16	5.1.1.16.2
RRC.ReEstabAtt	CC	RRC	16	5.1.1.17.1
RRC.ReEstabSuccWithUeContext	CC	RRC	16	5.1.1.17.2
RRC.ResumeAtt	CC	RRC	16	5.1.1.18.1
RRC.ResumeSucc	CC	RRC	16	5.1.1.18.2
RRC.ConnMean	SI	RRC	66	5.1.1.4.1
RRC.ConnMax	SI	RRC	78	5.1.1.4.2
RRC.InactiveConnMean	SI	RRC	66	5.1.1.4.3
RRC.InactiveConnMax	SI	RRC	78	5.1.1.4.4

Table II: 3GPP Key Performance Measurement (KPM) [8] telemetry reproduced in TELERAN.

IV. Security Applications

Rogue Base Station Detection. Rogue base stations, such as IMSI catchers, threaten cellular security by impersonating legitimate infrastructure to intercept traffic or disrupt service. Detection typically relies on identifying protocol-level inconsistencies at the cellular control plane [25]. Specifically, RRC Measurement Reports (e.g., RSRP, RSRQ) used for periodic mobility management of UEs offer a strategic vantage point for detection [24, 26]. By aggregating these reports, the network can detect anomalies like implausible signal gradients, inconsistent neighbor relationships, or impossible cell locations. Cross-UE correlation and temporal analysis further allow the system to distinguish malicious infrastructure from benign radio fluctuations.

Protocol-based Attack and Anomaly Detection. Protocol-based detection focuses on identifying attacks and misbehavior by monitoring control-plane message sequences and state transitions in cellular protocols. Table I lists the security telemetry that is captured by the TELERAN implementation, which have been used by prior studies to detect runtime cellular control plane attacks [27, 33–35]. For example, tracking temporal RRC and NAS message IDs enables the detection of out-of-order message sequences, such as signaling storms initiated with fabricated RRC connection requests [22]. Additionally, monitoring ciphering and integrity protection algorithms allows the system to identify runtime security violations, such as bidding-down or algorithm downgrade attacks targeting specific UEs.

Key Performance Monitoring. Key Performance Monitoring (KPM), standardized in 3GPP TS 28.552 [8], defines a set of performance metrics for monitoring RAN behavior, including connection establishment success rates, re-establishment attempts, and connection duration statistics. While originally intended for performance management and troubleshooting, these metrics provide valuable security signals. Sudden shifts in connection failures, abnormal increases in re-establishment attempts, or divergence between expected and observed success ratios may indicate ongoing attacks, misconfigurations, or rogue infrastructure interference. Integrating KPM with security analytics enables lightweight, operator-friendly anomaly detection that complements fine-grained protocol and radio-level monitoring. Table II summarizes the KPM telemetry that is captured by the TELERAN implementation

V. Conclusion and Future Work

We presented TELERAN, a novel framework addressing the fundamental telemetry gap in OpenRAN. By decoupling packet inspection from proprietary vendor implementations, TELERAN provides a fully vendor-agnostic, protocol-level visibility layer without modifications to existing source code. We envision three primary directions for future research:

- **Encrypted Traffic Analysis:** While TELERAN currently operates on cleartext interfaces, production 6G networks heavily utilize encryption. For kernel-level encryption (e.g., IPSec) [10, 11], advanced eBPF hooks at the Traffic Control (TC) layer can be utilized to intercept packets before encryption. For user-space encryption (e.g., PDCP and NAS layers [12, 13]), it requires instrumentation at cryptographic libraries to capture plaintext data before encryption or after decryption, such as via eBPF's `uprobes` and other userspace monitoring solutions [37].
- **Privacy and Security Hardening.** Since TELERAN exports plain-text control-plane telemetry, future work will focus on hardening the kernel-to-user space data path. While the current prototype relies on native Linux restrictions to protect eBPF maps, further integration of Trusted Execution Environments (TEE) could enforce strict isolation. These mechanisms would ensure that sensitive telemetry remains inaccessible to unauthorized processes, granting exclusive access to the authenticated TELERAN user-space agent.
- **Expansion to Open Fronthaul:** We intend to extend our protocol coverage beyond the F1 and NG interfaces to include the Open Fronthaul (OFH) [9]. This involves developing parsers for eCPRI and precise Ethernet-based transport protocols, enabling TELERAN to detect fronthaul-specific threats and synchronization anomalies.

Acknowledgment

This research was supported by a Small Business Innovation Research (SBIR) Phase I award N6893625C0023 from the Naval Air Warfare Center, the NSF convergence accelerator program under award ITE-2326882, and CNS-2112471.

REFERENCES

[1] Sdran. https://docs.sd-ran.org/master/introduction.html.

[2] O-ran.wg1.tr.decoupled-smo-architecture-r004-v03.00: O-ran decoupled smo architecture 3.0, October 2024.

[3] O-ran.wg3.ts.e2ap-r004-v08.00: O-ran e2 application protocol (e2ap) 8.0, October 2025.

[4] O-ran.wg3.ts.e2sm-kpm-r004-v07.00: O-ran e2 service model (e2sm) kpm 7.0, October 2025.

[5] O-ran.wg3.ts.e2sm-rc-r004-v09.00: O-ran e2 service model (e2sm), ran control 9.0, October 2025.

[6] O-ran alliance. https://www.o-ran.org/, Jan 2026.

[7] oai / openairinterface5g. https://gitlab.eurecom.fr/oai/openairinterface5g, Janurary 2026.

[8] 3GPP. 5g performance measurements. https://www.3gpp.org/DynaReport/28552.htm, Janurary 2026.

[9] 3GPP. Ng-ran architecture description. http://www.3gpp.org/DynaReport/38401.htm, Janurary 2026.

[10] 3GPP. Ng-ran f1 application protocol (f1ap). http://www.3gpp.org/DynaReport/38473.htm, Janurary 2026.

[11] 3GPP. Ng-ran; ng application protocol (ngap). http://www.3gpp.org/DynaReport/38413.htm, Janurary 2026.

[12] 3GPP. Non-access-stratum (nas) protocol for evolved packet system (eps). http://www.3gpp.org/DynaReport/24301.htm, Janurary 2026.

[13] 3GPP. Radio resource control (rrc). http://www.3gpp.org/DynaReport/38331.htm, Janurary 2026.

[14] Raphael Cannatà, Haoxin Sun, Dan Mihai Dumitriu, and Haitham Hassanieh. Towards seamless 5g open-ran integration with webassembly. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pages 121–131, 2024.

[15] eBPF. ebpf - introduction, tutorials & community resources. https://ebpf.io/.

[16] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. Taking 5g ran analytics and control to a new level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2023.

[17] The Linux Foundation. Open centralized unit distributed unit (ocudu). https://ocudu.org, Janurary 2026.

[18] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. srslte: An open-source platform for lte evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, pages 25–32, 2016.

[19] Tcpdump group. Home — tcpdump & libpcap. https://www.tcpdump.org/.

[20] Jun-Hong Huang, Shin-Ming Cheng, Rafael Kaliski, and Cheng-Feng Hung. Developing xapps for rogue base station detection in sdr-enabled o-ran. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2023.

[21] Ahan Kak, Van-Quan Pham, Huu-Trung Thieu, and Nakjung Choi. Hexran: A programmable multi-rat platform for network slicing in the open ran ecosystem. *arXiv preprint arXiv:2304.12560*, 2023.

[22] Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. Touching the untouchables: Dynamic security analysis of the lte control plane. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1153–1168. IEEE, 2019.

[23] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. {EdgeRIC}: Empowering real-time intelligent optimization and control in {NextG} cellular networks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1315–1330, 2024.

[24] Zhenhua Li, Weiwei Wang, Christo Wilson, Jian Chen, Chen Qian, Taeho Jung, Lan Zhang, Kebin Liu, Xiangyang Li, and Yunhao Liu. Fbs-radar: Uncovering fake base stations at scale in the wild. In *NDSS*, 2017.

[25] Kazi Samin Mubasshir, Imtiaz Karim, and Elisa Bertino. Gotta detect'em all: Fake base station and multi-step attack detection in cellular networks. In *Proceedings of the 34th USENIX Security Symposium*, 2025.

[26] Prajwol Kumar Nakarmi, Mehmet Akif Ersoy, Elif Ustundag Soykan, and Karl Norrman. Murat: Multi-rat false base station detector. *arXiv preprint arXiv:2102.08780*, 2021.

[27] Alessio Scalingi, Salvatore D'Oro, Francesco Restuccia, Tommaso Melodia, Domenico Giustiniano, et al. Det-ran: Data-driven cross-layer real-time attack detection in 5g open rans. In *IEEE International Conference on Computer Communications*, pages 1–10, 2024.

[28] Robert Schmidt, Mikel Irazabal, and Navid Nikaein. Flexric: An sdk for next-generation sd-rans. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, pages 411–425, 2021.

[29] Chuanhao Sun, Ujjwal Pawar, Molham Khoja, Xenofon Foukas, Mahesh K Marina, and Bozidar Radunovic. Spotlight: Accurate, explainable and efficient anomaly detection for open ran. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 923–937, 2024.

[30] the-tcpdump group. libpcap: the libpcap interface to various kernel packet capture mechanism. https://github.com/the-tcpdump-group/libpcap.

[31] vlm. asn1c: The asn.1 compiler. https://github.com/vlm/asn1c.

[32] Haoran Wan, Xuyang Cao, Alexander Marder, and Kyle Jamieson. Nr-scope: A practical 5g standalone telemetry tool. In *Proceedings of the 20th International Conference on emerging Networking EXperiments and Technologies*, pages 73–80, 2024.

[33] Haohuang Wen, Phillip Porras, Vinod Yegneswaran, Ashish Gehani, and Zhiqiang Lin. 5g-spector: An o-ran compliant layer-3 cellular attack detection service. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS'24)*, San Diego, CA, February 2024.

[34] Haohuang Wen, Phillip Porras, Vinod Yegneswaran, and Zhiqiang Lin. A fine-grained telemetry stream for security services in 5g open radio access networks. In *Proceedings of the 1st International Workshop on Emerging Topics in Wireless*, pages 18–23, 2022.

[35] Haohuang Wen, Prakhar Sharma, Vinod Yegneswaran, Phillip Porras, Ashish Gehani, and Zhiqiang Lin. 6g-xsec: Explainable edge security for emerging openran architectures. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pages 77–85, 2024.

[36] Wireshark. tshark. https://www.wireshark.org/docs/man-pages/tshark.html.

[37] Yusheng Zheng, Tong Yu, Yiwei Yang, Yanpeng Hu, Xiaozheng Lai, Dan Williams, and Andrew Quinn. Extending applications safely and efficiently. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*, pages 557–574, 2025.