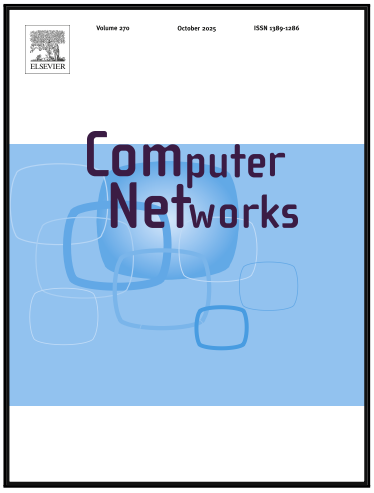


How to Poison an xApp: Dissecting Backdoor Attacks to Deep Reinforcement Learning in Open Radio Access Networks

Andrea Lacava, Stefano Maxenti, Leonardo Bonati, Salvatore D'Oro, Alina Oprea, Tommaso Melodia, Francesco Restuccia

PII: S1389-1286(25)00693-0
DOI: https://doi.org/10.1016/j.comnet.2025.111727
Reference: COMPNW 111727

To appear in: *Computer Networks*

Received date: 30 April 2025
Revised date: 9 July 2025
Accepted date: 17 September 2025

Please cite this article as: Andrea Lacava, Stefano Maxenti, Leonardo Bonati, Salvatore D'Oro, Alina Oprea, Tommaso Melodia, Francesco Restuccia, How to Poison an xApp: Dissecting Backdoor Attacks to Deep Reinforcement Learning in Open Radio Access Networks, *Computer Networks* (2025), doi: https://doi.org/10.1016/j.comnet.2025.111727

How to Poison an xApp: Dissecting Backdoor Attacks to Deep Reinforcement Learning in Open Radio Access Networks

Andrea Lacava[a,*], Stefano Maxenti[a], Leonardo Bonati[a], Salvatore D'Oro[a], Alina Oprea[b], Tommaso Melodia[a], Francesco Restuccia[a]

[a] *Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, USA*
[b] *Khoury College of Computer Sciences, Northeastern University, Boston, MA, USA*

Abstract

The development of Open Radio Access Network (RAN) cellular systems is being propelled by the integration of Artificial Intelligence (AI) techniques. While AI can enhance network performance, it expands the attack surface of the RAN. For instance, the need for datasets to train AI algorithms and the use of open interface to retrieve data in real time paves the way to data tampering during both training and inference phases. In this work, we propose MalO-RAN, a framework to evaluate the impact of *data poisoning* on O-RAN intelligent applications. We focus on AI-based xApps taking control decisions via Deep Reinforcement Learning (DRL), and investigate backdoor attacks, where tampered data is added to training datasets to include a backdoor in the final model that can be used by the attacker to trigger potentially harmful or inefficient pre-defined control decisions. We leverage an extensive O-RAN dataset collected on the Colosseum network emulator and show how an attacker may tamper with the training of AI models embedded in xApps, with the goal of favoring specific tenants after the application deployment on the network. We experimentally evaluate the impact of the SleeperNets and TrojDRL attacks and show that backdoor attacks achieve up to a 0.9 attack success rate. Moreover, we demonstrate the impact of these attacks on a live O-RAN deployment implemented on Colosseum, where we instantiate the xApps poisoned with MalO-RAN on an O-RAN-compliant Near-real-time RAN Intelligent Controller (RIC). Results show that these attacks cause an average network performance degradation of 87%.

Keywords: Open RAN, 5G, AI, Adversarial AI, DRL

1. Introduction

Cellular networks based on 5th Generation (5G) and beyond technologies are designed to support heterogeneous use cases on an unprecedented scale, requiring automated control and optimization of network capabilities tailored to the needs of users and operators. In recent years, the Open Radio Access Network (RAN) paradigm has emerged, whose specifications introduce an open architecture

*Corresponding author
Email addresses: a.lacava@northeastern.edu (Andrea Lacava), maxenti.s@northeastern.edu (Stefano Maxenti), l.bonati@northeastern.edu (Leonardo Bonati), s.doro@northeastern.edu (Salvatore D'Oro), a.oprea@northeastern.edu (Alina Oprea), t.melodia@northeastern.edu (Tommaso Melodia), f.restuccia@northeastern.edu (Francesco Restuccia)

with abstractions that enable closed-loop control and allow real-time monitoring and intelligent optimization of the network based on data collected from the RAN. This adds intelligence and programmability in the loop through the introduction of RAN Intelligent Controllers (RICs) that host Artificial Intelligence (AI) applications, namely xApps and rApps, working at different timescales to optimize the network performance [1].

This transition from monolithic RAN deployments to the O-RAN system is being managed by the O-RAN ALLIANCE, which is defining an architecture for the future generations of cellulars networks via the release of specifications and use cases. Through their embedding in these novel and modular O-RAN applications, AI will play a pivotal role in future generations of virtualized and programmable cellular networks [2]. Indeed, applications will be able to extract data from different points of the network through standardized, open and programmable interfaces, and to perform network optimization and control at scale and granularity unseen in previous cellular generations [3].

Existing Issues. The openness and disaggregation of the RAN unavoidably extends its security issues. For example, it has been shown that spoofing and time synchronization attacks can cause severe RAN performance degradation [4]. In addition, xApps can be exposed to Adversarial Machine Learning (AML) attacks where *data poisoning* of RAN measurements could seriously disrupt their performance.

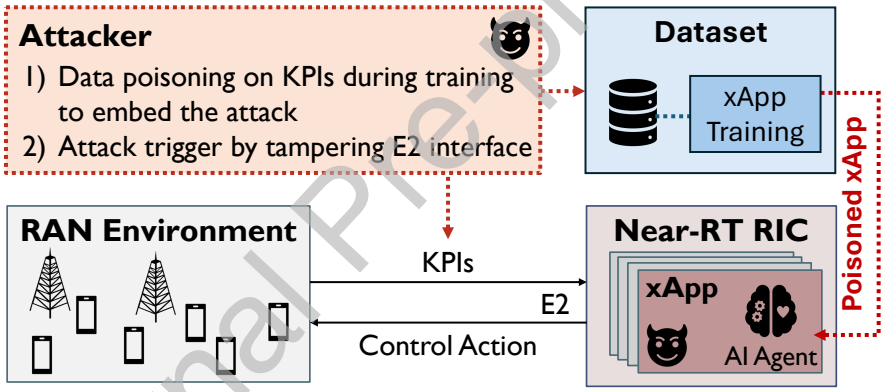


Figure 1: Overview of attacks to O-RAN xApps.

As an example, Figure 1 shows the structure of a data poisoning attack in O-RAN executed at training time. The attack introduces a backdoor that can be triggered when the xApp is used to control the RAN [5]. To this end, the attacker introduces imperceptible modifications in the dataset, undermining the correct behavior of the Machine Learning (ML) model and of the system. The anomalous behavior is triggered at run-time by tampering with the data transmitted by the RAN to the RIC (and to the xApp) via the O-RAN E2 interface. Upon receiving the trigger, the xApp produces a crafted control action, which is then sent to the RAN and enforced therein.

These attacks are made even more effective by downloading compromised xApps from an application marketplace, which exposes the entire O-RAN ecosystem to similar attacks [6, 7, 8]. One relevant example is that of an xApp controlling radio resource allocation policies via a Deep Reinforcement Learning (DRL) agent. Although the xApp should guarantee fairness among all User Equipments (UEs), poisoning attacks can cause unfair advantages to selected UEs by prioritizing the

subscribers of a specific service provider or by degrading the performance of those of competitors.

Technical Challenges. Although the effectiveness of poisoning attacks has been studied in traditional AI/ML settings [9], their impact on O-RAN is still an open question. Most importantly, a comprehensive framework for comparing different poisoning attacks while being able to monitor RAN performance to identify the attacks is still missing. Developing such framework is complicated by the fact that production-level RAN cannot be used as any performance degradation would affect mobile subscribers.

To fill this gap, in this work we propose MalO-RAN, a novel framework to design, implement and test the robustness of AI models against backdoor and poisoning attacks in O-RAN. Starting from the monitoring and control capabilities offered by the RICs and their intelligent applications, we consider *poisoned control loops*, and show how a malicious entity can influence the input of an optimization agent to force an arbitrary policy designed by the attacker as the control action produced by the agent. We use a publicly available dataset collected on the Colosseum network emulator [10] to train and evaluate existing attacks such as TrojDRL [11] and SleeperNets [12] against DRL-based agents in O-RAN. This dataset leverages the capabilities of Colosseum to faithfully reproduce realistic RAN deployments with multiple Base Stations (BSs) and UEs, and network and channel conditions. Moreover, we demonstrate for the first time the effects of poisoned xApps on the performance of a live O-RAN deployment instantiated on the Colosseum platform. To the best of our knowledge, MalO-RAN is the first framework to enable the evaluation of the impact of poisoning attacks on RAN performance.

Main Contributions. The main contributions of this paper are summarized as follows:

- We propose, design and prototype MalO-RAN, an open-source framework to evaluate the feasibility, effectiveness and transferability of poisoning attacks against O-RAN AI/ML applications. We build upon the TrojDRL and SleeperNets attacks [11, 12] to train AI/ML models aiming at manipulating the allocation of network slicing resources via malicious xApps.
- We extend the SleeperNets attack vector to improve stealthiness by introducing complex patterns—such as combinations of multiple Key Performance Measurements (KPMs), i.e., more than a single parameter—embedded in the model during training and capable of triggering the attack during deployment.
- We evaluate our framework using a large and realistic dataset collected on Colosseum, the world's largest O-RAN digital twin with hardware in the loop [10].
- We show the effectiveness of the considered attacks applied to the O-RAN domain, which achieve more than 0.9 success rate in the case of SleeperNets.
- We demonstrate for the first time the impact that an xApp poisoned with such attacks has on a live O-RAN deployment implemented on Colosseum, showing that the poisoned xApp causes average network performance degradations of 87% in the user throughput.
- We make the MalO-RAN framework publicly available to foster and facilitate research toward increased robustness of O-RAN.[1]

[1] MalO-RAN is available at https://github.com/wineslab/mal-o-ran.

Although the present study evaluates SleeperNets and TrojDRL, MalO-RAN is purposely architected to accommodate any future adversarial method through the same modular pipeline. The remainder of this paper is organized as follows. Section 2 provides some background knowledge on O-RAN. Section 3 surveys related work on adversarial ML, with a specific focus on its application to 5G-and-beyond cellular systems. Section 4 describes the MalO-RAN framework that we developed to apply DRL attacks to O-RAN networks, as well as how to integrate attacks into MalO-RAN. Section 5 describes the O-RAN dataset that we used in this work, and provides details related to data poisoning for AI/ML training. Section 6 experimentally evaluates the effectiveness of the considered attacks when applied to O-RAN. Finally, Section 7 draws our conclusions.

2. An Introduction to O-RAN Cellular Networks

The Open RAN paradigm, in its embodiment from the O-RAN ALLIANCE, adopts a disaggregated approach to network deployment by separating traditionally integrated elements into smaller, modular components that communicate over open and standardized interfaces [1]. This shifts the network architecture from a vertically integrated setup to a horizontal one, prioritizing scalability and programmability through softwarized components with support for multi-vendor deployments. Base station functionalities are now virtualized and split across different network elements, namely the Central Unit (CU) (implementing higher-layer protocol stack functionalities), the Distributed Unit (DU) (higher Physical (PHY) layer, Medium Access Control (MAC), and Radio Link Control (RLC) functionalities), and the Radio Unit (RU) (lower PHY and Radio Frequency (RF) functionalities), for enhanced flexibility and programmability.

These network elements can be controlled in software via programmable Application Programming Interfaces (APIs), which paves the way to the RICs, one of the main innovations introduced by O-RAN. The RICs come in different flavors and oversee the operations of the RAN at different timescales. Specifically, the Near-real-time (or Near-RT) RIC acts at granularities ranging from 10 ms to 1 s, and implements closed-loop control via AI and ML applications named xApps; the Non-real-time (or Non-RT) RIC, instead, is part of the Service Management and Orchestration (SMO) framework, and acts at timescales above 1 s via applications named rApps. The O-RAN next-Generation Research Group (nGRG) is also investigating and evaluating the introduction and use of a novel type of real-time applications named dApps, which can be deployed directly at the CU/DU to enable sub-10 ms network control [13, 14], and thus support tighter control loops at the edge and potentially extend the attack surface also in the physical domain. Overall, common use cases enabled by O-RAN include optimization of network slicing and scheduling strategies, traffic steering, intelligent beamforming and mobility management, and forecasting of network performance based on reported RAN Key Performance Indicators (KPIs), to name a few.

On the one hand, this novel approach to cellular networking promotes flexible control and scalability of the network elements, paves the way toward sustainable RAN deployments and improved performance and Quality of Service (QoS). On the other hand, however, it also introduces an additional degree of complexity in managing and coordinating the novel interfaces and distributed components [1], which inevitably increases the attack surface of the network [4].

3. Related Work

With the recent mainstream adoption of AI technologies, adversarial machine learning (AML) has emerged as a critical and rapidly growing area of research. AML focuses on understanding and

mitigating vulnerabilities in AI models, particularly when exposed to maliciously crafted inputs. These vulnerabilities pose significant security risks across various domains, including computer vision, natural language processing, and network systems. Consequently, an increasing body of research has sought to identify, analyze, and defend against adversarial attacks to ensure the robustness and reliability of AI systems. In this section, we aim to survey works related to our research. Specifically, Section 3.1 discusses AML works in general, while Section 3.2 illustrates attacks applied to the wireless and O-RAN ecosystem.

3.1. Adversarial Machine Learning Attacks

Prior work focused on AML attacks for classification tasks at inference time, as well as data poisoning attacks at training time [15, 16, 11, 17, 12]. Inference-time attacks are conducted by slightly manipulating the input of a ML model to include adversarial data points. In its basic form, this can be achieved by adding noise to the input to generate misclassifications and even to steer the output toward a target output (e.g., a certain class, label, or control action). A more advanced version of this attack aims to do the same by introducing specific and optimized perturbations in the Neural Network (NN) of the model. Examples are Fast Gradient Sign Method (FGSM) [15] and Projected Gradient Descent (PGD) [16]—usually targeting image classification—that compute the gradient of the Deep Neural Network (DNN) and produce a perturbation that maximized its loss function.

Data poisoning attacks focus on modifying the training dataset to alter the expected behavior of the trained model. For instance, [11] proposes a trigger-based attack that poisons the training of DRL models to introduce a backdoor capable of altering the output of the agent when a certain trigger is fed in input to the model. Similarly, [17] proposes BadRL, an algorithm that computes the optimal attack strategy to determine when and how backdoors are introduced during the training phase to maximize their effectiveness, while reducing the probability of being detected. The work closest to ours is SleeperNets [12], where the authors show that previous attacks have limitations concerning their static reward poisoning techniques, and they lack former analysis of the proposed attacks. For the first issue, they describe how static attacks cannot adapt to various Markovian Decision Processes (MDPs) or algorithms. For the second one, instead, they highlight how focusing only on inner-loop attacks limits the information the attacker has access to during training, i.e., the information on the completed episodes to adapt the poisoning strategy. SleeperNets overcomes these limitations by introducing an outer-loop threat model, where an adversary manipulates the agent reward and state observation after each episode.

Other works investigating the effect of adversarial ML attacks on DRL models include [18, 19, 20]. Specifically, [18] manipulates the observation space of the model showing that DNN trained through DRL are susceptible to adversarial attacks akin to their supervised learning counterpart. Similarly, [19] demonstrates how both naive and more targeted attacks leveraging gradient information can impact the behavior of DRL agents. Work [20] introduces two types of attacks: the Critical Point Attack, and the Antagonist Attack. The former targets domain-specific models to steer future steps toward the goal of the attacker; the latter applies to domain-agnostic models and instructs the adversary on when and how to add the perturbations needed for the attack.

Additional attacks found in the literature include the introduction of a bi-level optimization problem, where an outer loop identifies inputs to maximize the loss function of the training dataset, while an inner one retrains on the perturbed dataset [5]. However, this requires complete access to the model and dataset. In [21], the authors focus on finding the optimal training-set attack for support vector machines, and logistic and linear regression. However, this work does not investigate

the applicability of the proposed technique to Reinforcement Learning (RL) problems. Finally, a comprehensive survey on DRL attacks can be found in [9].

Differently from the work above, which is generically applied to domains such as image classification for computer vision, and robotic control problems, our work focuses on adversarial AI attacks targeting control applications (e.g., xApps) operating in the context of O-RAN.

3.2. Attacks to Wireless and O-RAN Networks

In the context of wireless communications, a wide body of literature focuses on the physical layer of the network. For example, the authors of [22] study a generalized wireless adversarial machine learning problem where attack vectors are injected directly on waveforms. The authors of [23] deal with the issue of synchronization between a benign and evil user on the network, and the effect on the channel over the adversarial attack. These attacks are powerful and generic, but they lack direct applicability for next-generation and O-RAN networks.

A smaller subset of literature works focus specifically on the O-RAN ecosystem. An overview of the organization of O-RAN ALLIANCE WG-11, focused on security, is provided in [24], highlighting the current state of security and potential threat models and attack vectors. A taxonomy for identifying adversarial ML attacks in O-RAN and categorizing them is provided in [5], which thoroughly analyzes different kinds of attacks from poisoning, to manipulation, and perturbation of the data. In [25], authors show the effectiveness of the PGD attack applied to the DRL models of O-RAN xApps. The authors highlight, however, that these attacks can be partially mitigated through robust training. Another recent publication [26] describes an attack targeted to DRL agents for resource allocation, showing how manipulating UE measurements (from compromised users or signal jammers) reduces the network performance and increases latency, causing relevant disruption in Vehicle-To-Everything (V2X) communication.

Some works investigate the lack of encryption and authentication of the O-RAN interfaces, which can be exploited by AML attacks. Motivated by this, [4] analyzes the trade-offs between encryption mechanisms and latency on the O-RAN interfaces among Near-RT RIC and CU/DU. Similarly, [27] demonstrates the feasibility of redirecting RAN traffic to malicious xApps, as well as data tampering. This is made possible by the lack of authentication among BS, Near-RT RIC, and xApps.

The authors of [28] focus on classification attacks in the O-RAN framework for spectrum sensing in xApps, showing the implementation of FGSM attacks against the RF spectrum and KPMs. They also propose a distillation-technique to reduce the efficacy of the attacks. However, they do not investigate DRL. A separate research line focuses on detection of attacks. The authors of [29], for example, propose an xApp for detecting and mitigating backdoor O-RAN attacks based on Long Short Term Memory (LSTM) autoencoder considering historical data. While this is a promising work, it does not clearly identify which kind of adversarial machine learning attacks it has been tested against and only focuses on Internet of Things (IoT) datasets. Finally, [30] analyzes methodologies for attacking a DRL model trained to assign Physical Resource Blocks (PRBs) to cellular UEs , whereas a recent work [31] describes a novel framework to mitigate the effect of various adversarial attacks applied to the same resource allocation problem.

In this paper, we propose MalO-RAN, a unified and extensible framework to design, evaluate, and compare generic adversarial ML attacks on the O-RAN ecosystem, with the possibility of testing the poisoned models over the Colosseum testbed in a live O-RAN deployment.

4. MalO-RAN and Attack Integration

In this section, we introduce our proposed MalO-RAN framework, which facilitates the design, prototyping, and evaluation of attacks on DRL agents within the O-RAN ecosystem. Specifically, Section 4.1 outlines the architecture of MalO-RAN, Section 4.2 details the attack models considered, and Section 4.3 explains how MalO-RAN can be used to integrate and evaluate DRL-based adversarial attacks in the O-RAN environment.

4.1. Architecture of MalO-RAN

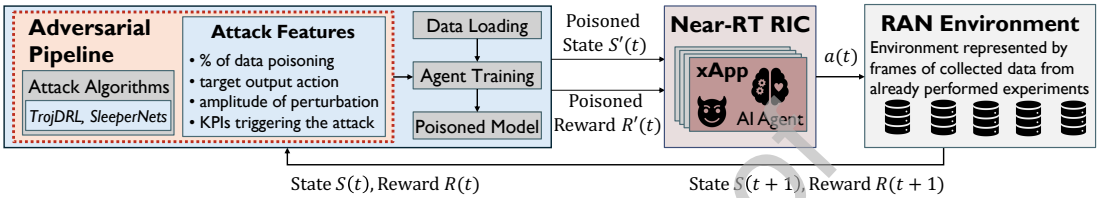


Figure 2: Overview of the architectural blocks and procedures of MalO-RAN to train a poisoned xApp offline using pre-existing data.

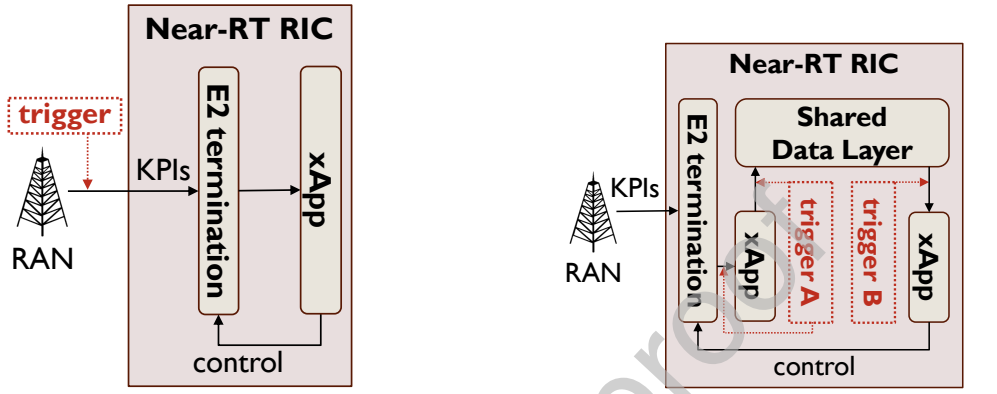
Figure 2 shows the architecture of MalO-RAN, which is composed of (i) the RAN Environment; (ii) the Adversarial Pipeline; and (iii) the Near-RT RIC. The RAN Environment concerns a RAN deployed with softwarized 5G protocol stacks, such as OpenAirInterface (OAI) and srsRAN, that generate metrics and KPIs. These are stored in a database or data lake, and are then used to train O-RAN AI/ML agents. In our case, we leverage the O-RAN dataset collected in [8] and described in Section 5. The dataset contains KPIs collected from live experiments performed with the srsRAN cellular protocol stack deployed on Colosseum.

The Adversarial Pipeline is in charge of poisoning and training the AI/ML models that are onboarded on the xApps. As shown in Figure 2, it includes two main elements. These are the data-loading module, and the agent training module, which eventually produce the poisoned AI/ML model. The data-loading module is used to load and pre-process the data coming from the RAN Environment. The training module, instead, trains the DRL model and introduces the perturbations needed to poison the processed data, thus generating a poisoned model. The model trained in this way can be then embedded into O-RAN applications, e.g., xApps, and used to perform a closed-loop control susceptible to backdoor attacks on a live RAN. In this work, we implement the Adversarial Pipeline by extending the open-source SleeperNets framework [12], which is based on PyTorch and Gymnasium [32, 33]. Our pipeline exposes generic hooks so that new poisoning, evasion, or hybrid attacks can be registered without altering the core data flow. Moreover, we train our DRL agent offline, as mandated by the O-RAN AI/ML specifications, which require data-driven applications to undergo an offline training phase before their deployment not to cause outages or disruptions to the RAN [34]. To support training and attack operations over the O-RAN dataset, we created a custom Gymnasium offline environment dedicated to connecting the adversarial pipeline with the dataset. This allows standardized access for testing different attack combinations and reproducibility of experiments.

Finally, the Near-RT RIC hosts the trained AI/ML model in the form of an xApp. The xApps poisoned with MalO-RAN can be used to send malicious control actions to the CU/DU when

triggered by specifically crafted RAN KPIs received as input via the O-RAN E2 interface. For instance, this could represent the case of a tenant unknowingly onboarding a poisoned application from an xApp catalog. In this work, we leverage the O-RAN-compliant Near-RT RIC provided by OpenRAN Gym [35] and instantiated on Colosseum.

4.2. Modeling the Poisoning Attacks



(a) Direct Trigger

(b) Shared Data Layer (SDL) Trigger

Figure 3: Considered attack models and triggers.

Figure 3 shows the two attack models and triggers that we consider: attack model with direct trigger, and attack model with SDL trigger.

The former is depicted in Figure 3a and consists in tampering with RAN KPIs sent by the CU/DU to the Near-RT RIC (and xApp) through the O-RAN E2 interface. This is the case, for instance, of a malicious entity that does not have direct access to the Near-RT RIC, but has knowledge of the xApps and AI models running on it, and wants to trigger an already embedded attack in some other way (e.g., by poisoning an open dataset). The latter is shown in Figure 3b and can have attack triggers of two different types, namely trigger A and B in the figure. Trigger of type A occurs when the xApp or an attacker with direct access to the Near-RT RIC applies a trigger to legitimate data received from the RAN, and pushes it to SDL—a database located in the Near-RT RIC and shared across multiple xApps—where benign xApps, then, pull data from to produce a RAN control action. Trigger of type B, instead, happens when the trigger is applied to the KPIs extracted from SDL by benign xApps, e.g., to perform inference or control tasks.

We consider two kind of attackers, as defined in [9, 11]: *strong* and *weak* attacker. The strong attacker corresponds to a scenario in which the training of the agent is outsourced to an external service provider (e.g., an xApp vendor that publishes a trained xApps to a common marketplace or catalog). In this case, the attacker acts on the provider side, and possesses enhanced tampering capabilities and full access to all the details of the model. This scenario is highly relevant in O-RAN where the marketplace model will be extensively used by operators to purchase applications from third-party providers.

The weak attacker, instead, corresponds to a scenario in which the agent is trained in a compromised environment, and the attacker only has access to the relevant data and KPIs but not to

the model itself. In this case, the attacker can only manipulate the KPIs input to the agent. While this attacker has been included in MalO-RAN, its feasibility in O-RAN systems is very limited. Indeed, O-RAN requires models to be trained offline, and online training is only limited to cases of fine-tuning of pre-trained models, which makes the weak attacker less likely to happen. For this reason, the evaluation of weak attackers in practical deployment is left for future studies, and here we focus on the strong attacker.

4.3. Integration of Attacks in MalO-RAN

The purpose of MalO-RAN is to allow users to (i) design and evaluate the impact of novel adversarial attacks on O-RAN systems, and (ii) determine how adversarial attacks designed for other domains (e.g., computer vision, robotics, etc.) can be transferred to the O-RAN ecosystem, as well as to its applications and use cases. For this reason, MalO-RAN follows a modular design, where users can extend the Adversarial Pipeline discussed in Section 4.1 to include novel attacks and exploit them to poison the data coming from the RAN environment. The newly poisoned data can, then, be used to train adversarial agents, deploy them on the Near-RT RIC in the form of xApps, and evaluate their impact on the RAN environment via MalO-RAN.

We integrate two state-of-the-art poisoning and backdoor attacks in MalO-RAN: TrojDRL [11] and SleeperNets [12]. MalO-RAN exposes the configurable features of the above algorithms, providing external APIs for users to fine-tune attack parameters before running the training. This enables them to specify key attack parameters such as the percentage of data poisoning to be performed at each iteration, the target action the model should produce given a specific input trigger, and a list of the RAN KPIs to alter during the attack. Moreover, we enhance and extend the SleeperNets codebase [36]—which allows users to alter single features only—to support the concurrent modification of multiple parameters. Starting from the original code, we have rewritten the `SingleValuePoison` module to create a new `MultiValuePoison` class, which enables the concurrent substitution of multiple input features by specifying a list of indices and corresponding values to be applied at attack time. This allows MalO-RAN to generate more complex poisoning attacks, facilitating the evaluation of attacks targeting multiple KPIs, and making it possible to understand how increasingly complex attack strategies affect the RAN performance. This functionality is well suited to O-RAN applications and use cases, as RAN datasets include a large number of features, as will be discussed in Section 5.1.

The list of parameters that can be fine-tuned with MalO-RAN is summarized in Table 1. Relevant parameters include: (i) the attack algorithm (to be chosen between TrojDRL and SleeperNets); (ii) the poisoning budget β, which specifies the specifies the fraction of state observations they can poison in training, i.e., the percentage of poisoned observations with respect to the total number of observations in the dataset; (iii) the reward perturbation magnitude κ, to alter the value of the agent reward; (iv) the weighting factor α, used to weight rewards in SleeperNets; (v) the target action to execute upon triggering the attack; (vi) the attacker model, chosen between strong and weak attacker (see Section 4.2); (vii) the number of epochs of the training phase; and (viii) the learning rate of the Adam optimizer. The reward perturbation during the training process depends on the specific attack algorithm used. In the case of TrojDRL, the perturbation is static—i.e., a fixed κ value is applied consistently throughout training, according to the poisoning budget β. In contrast, SleeperNets uses a dynamic reward mechanism that leverages post-trajectory hindsight to retroactively shape the reward signal. This allows the adversary to better reinforce the target behavior (e.g., executing a specific action when a trigger is present), thereby increasing the stealthiness and effectiveness of the backdoor. For a more detailed description of how these parameters

Table 1: Attack parameters and values used in evaluation.

Parameter	Values	Description
Attack algorithm	TrojDRL, SleeperNets	Backdoor attacks for DRL implemented in MalO-RAN
β	$0.005, 0.01, 0.1, 0.25$	Poisoning budget (% of times the attacker tries to poison)
κ	$0.1, 0.2, 0.5$	Absolute offset to change the agent reward
α	$0, 0.5, 1$	Weighting factor on the agent reward perturbation (SleeperNets only)
Target Action	0	Action forced by the attack trigger, i.e., 3 PRBs
Attacker model	Strong	Type of attack performed
Number of epochs	$150,000$	Number of training epochs
Learning rate	0.00025	Learning rate of the Adam optimizer

interact with each other during the attack, please refer to Algorithm 1 in the original SleeperNets paper [12].

5. O-RAN Datasets and Data Poisoning

In this section, we describe the O-RAN dataset that we use in this work, and the DRL agent used to validate the attack (Section 5.1), as well as the data poisoning techniques and methodology to enable backdoor attacks (Section 5.2).

5.1. Colosseum O-RAN Dataset and DRL Agent

We consider a publicly available large-scale dataset [8], used in several works to train agents for a variety of use cases ranging from optimization of scheduling policies and PRB allocation [37], to explainable AI in cellular networks [38]. The dataset has been collected on Colosseum, a large-scale wireless network emulator that leverages the OpenRAN Gym framework [35] to instantiate standard-compliant RANs through software such as srsRAN [39], as well as the O-RAN Software Community (OSC) Near-RT RIC and AI/ML xApps. It includes 73 hours of KPIs and metrics collected from seven BSs, each operating with a 10 MHz channel bandwidth (50 PRBs) and serving 42 UEs distributed within a 50-meter radius from their respective BS. Users are divided across three different network slices implemented by each BS, which serve them different traffic loads: Enhanced Mobile Broadband (eMBB) (constant-bit rate traffic at 4 Mbps), Machine-type Communications (MTC) (Poisson-distributed traffic at 30 kbps), and Ultra Reliable and Low Latency Communications (URLLC) (Poisson-distributed traffic at 10 kbps). The dataset includes KPIs collected for different configurations of the BSs for what concerns the amount of PRBs allocated to each slice, as well as the scheduling policy of each slice, which can be chosen among the round-robin, waterfilling, and proportionally fair.

The metrics of the dataset, which are collected by the BSs at a UE-level granularity and aggregated every 250 ms, are summarized in Table 2. They include the downlink and uplink throughput and buffer occupancy, as well as Power Headroom Report (PHR), and the number of retransmissions, to name a few. Additionally, for each metric, the dataset includes details on the BS configuration (e.g., number of PRBs, scheduling policy, etc.), which can represent the action taken by an AI/ML agent embedded in an xApp and used to control the RAN. We perform the pre-processing

of the dataset to remove any malformed data, after which we obtain a cleaned-up dataset with 35,512,393 observations and more than 6 GB in size.

Although MalO-RAN is general and one can define diverse attack strategies and engineer different triggers and target actions, for the sake of illustration, in the following we focus on a specific use case. Specifically, we leverage this dataset to define the attack optimization problem, namely Problem NOP, where the goal of the attacker is to generate a trigger input that pushes the DRL agent to unfairly allocate fewer resources to UEs belonging to a slice under attack (e.g., all customers of a competitor). The problem is formulated as follows:

$$\max_{\mathbf{a}} \sum_{t=0}^{+\infty} \sum_{i=1}^{N} \frac{p_i^{ACT}(t)}{p_i^{REQ}(t)} \tag{NOP}$$

$$\text{subject to} \sum_{i=1}^{N} p_i^{ACT}(t) \leq P, \quad \forall t = 0, \ldots, +\infty \tag{1}$$

$$p_i^{ACT}(t) = \sum_{u=1}^{U} \sum_{i=1}^{N} \gamma_{u,i}\, p_u^{ACT}(t), \forall t = 0, \ldots, +\infty \tag{2}$$

$$p_i^{REQ}(t) = \sum_{u=1}^{U} \sum_{i=1}^{N} \gamma_{u,i}\, p_u^{REQ}(t), \forall t = 0, \ldots, +\infty \tag{3}$$

where $p_i^{REQ}(t)$ and $p_i^{ACT}(t)$ are the amount of PRBs requested and actually allocated to the UEs of slice i by the BS over time t, respectively, with $p_u^{REQ}(t)$ and $p_u^{ACT}(t)$ those of the individual UE u. $\mathcal{U}$ is the set of U UEs served by the BS, with $U = |\mathcal{U}|$, and $\gamma_{u,i} \in [0, 1]$ is a binary variable equal to 1 if UE u belongs to slice i, and 0 otherwise. N is the number of slices implemented at the BS, while P is the PRB budget of the BS. In the case of the considered dataset, we have that $N = 3$ and $P = 50$. Our agent can control the system through a variable $\mathbf{a} = \{a(t) = (p_i^{ACT}(t))_{i=1,\ldots,N}\}_{t=0,\ldots,+\infty}$ that represents the amount of PRBs jointly allocated to the N slices where $a(t) \in \mathcal{A}$ is chosen among a finite set of actions $\mathcal{A}$, with $A = |\mathcal{A}|$ possible combinations contained in the dataset (in the case of our dataset, $A = 21$). These range from a minimum of 3 PRBs to a maximum of 42 PRBs for each slice, constrained by a maximum of P PRBs available at the BS. The objective of Problem NOP is to maximize the ratio between the number of PRBs allocated to the UEs and the number of PRBs they request based on their traffic demand. In general, this ratio represents a satisfaction indicator where a value of 1 means that the UE is fully satisfied, and values close to 0 mean that the demand of the UE is not satisfied. Differently from [8], we consider the case where the agent does not include any autoencoder for dimensionality reduction. Indeed, the autoencoder creates a latent representation of the input vector, which might prevent our study from deriving insights on how poisoned data affects the behavior of the agent. By removing the autoencoder from the xApp, we establish a direct mapping between the poisoned values and the input features to better analyze the influence of the poisoning on the observation state.

Problem NOP is not easy to solve as it requires to compute a control policy that maximizes UE satisfaction over a long period of time. Moreover, it assumes that (i) we have a closed-form model for capturing traffic demand and its relationship with PRB demand; (ii) we have perfect information on UE number and slice assignment; and (iii) we can predict future evolutions of the network. This makes directly solving Problem NOP impractical. For this reason, we follow a data-driven approach where we use the dataset from [8] to design an agent to solve Problem NOP.

Table 2: KPIs relative to the UEs served by the BS for the observation space. From the Col-O-RAN dataset [8].

Metric	Description
num_ues	Number of UEs connected to the BS
slice_prb	Number of PRBs assigned to each slice. This is the control action computed by the DRL agent
scheduling_policy	Policy used to schedule slice UEs, among round-robin, waterfilling, and proportionally fair scheduling policies
dl_mcs	Downlink Modulation and Coding Scheme
dl_n_samples	Number of downlink samples transmitted to the UE
dl_buffer	Occupancy, in byte, of the downlink buffer with data to transmit to the UE
tx_brate downlink	UE downlink throughput in Mbps
tx_pkts downlink	Number of downlink packets transmitted to the UE
dl_cqi	Downlink Channel Quality Indicator
ul_mcs	Uplink MCS
ul_n_samples	Number of uplink samples transmitted by the UE
ul_buffer	Occupancy, in byte, of the uplink buffer with data received from the UE
rx_brate uplink	UE uplink throughput in Mbps
rx_pkts uplink	Number of uplink packets transmitted by the UE
rx_errors uplink	Percentage of uplink reception errors
ul_sinr	Uplink Signal to Interference plus Noise Ratio
phr	Power Headroom Report
sum_requested_prbs	Sum of PRBs requested by the UE
sum_granted_prbs	Sum of PRBs allocated to the UE
ul_turbo_iters	Number of turbo iterations in the uplink

Specifically, we leverage an actor-critic NN with a shared feature extractor. The agent architecture includes two hidden layers with 64 neurons, each one of them with Rectified Linear Unit (ReLU) activation functions. The actor component of the NN generates action probabilities, while the critic estimates the state values. The input observations are normalized before being processed, and the model is optimized using the Adam optimizer with carefully tuned parameters to ensure stability and efficiency.

We use MalO-RAN to train our agent offline with the Proximal Policy Optimization (PPO) algorithm. The observation space S corresponds to a subset of KPIs reported by the BS (see Table 2). In Section 5.2 we will investigate how the design of the observation space (e.g., different subsets of KPIs) affect attack success rate and impact network performance. The reward function R corresponds to the utility function in Problem (NOP), and the action space is $\mathcal{A}$ such that, at each time instant t, the agents selects action $a(t)$, i.e., the PRB allocation for the attacked slice.

5.2. Data Poisoning for Backdoor Attacks

To gather insights on the feasibility and effectiveness of poisoning attacks in O-RAN systems, we designed the MalO-RAN framework to offer tools for an exhaustive evaluation of poisoning attacks under a variety of parameters and configurations. These include the ability to select the attack algorithm, the target KPIs to poison, the severity of the poisoning attack (i.e., the magnitude of the perturbation caused to KPI values), and the target output action to steer the agent toward whenever the trigger input is sent to the agent. Although MalO-RAN can be used to test any combination of parameters, the specific values considered in our evaluation are summarized in Table 1. These values were chosen to cover a wide range of attack scenarios. While not exhaustive,

they let us evaluate the impact of the different parameters on the Attack Success Rate (ASR) and attack detectability.

For the purpose of our evaluation, we consider the case where the attacker aims at generating poisoned data that results in the DRL agent selecting an action that penalizes a certain slice to produce UE dissatisfaction. Specifically, the attacker targets eMBB UEs and aims at reducing their quality of experience. Data analytics from the original dataset [8] show that users of the eMBB slice are the ones generating the highest demand and require the largest amount of PRBs to be satisfied. For this reason, we consider the case where the attacker aims at generating a poisoned input such that eMBB UEs get allocated with the minimum amount of PRBs among the combinations of PRB allocations available in the considered dataset (i.e., 3 PRBs, as discussed in Section 5.1).

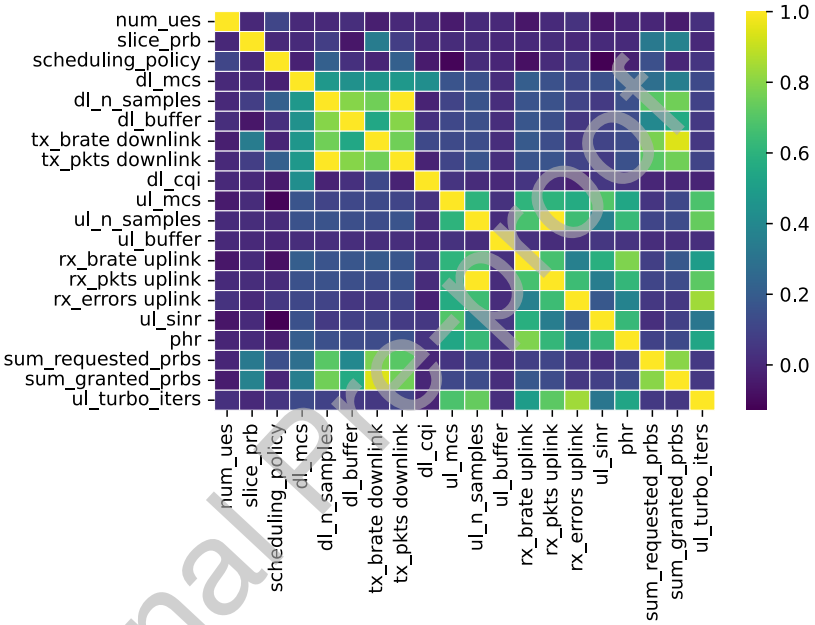


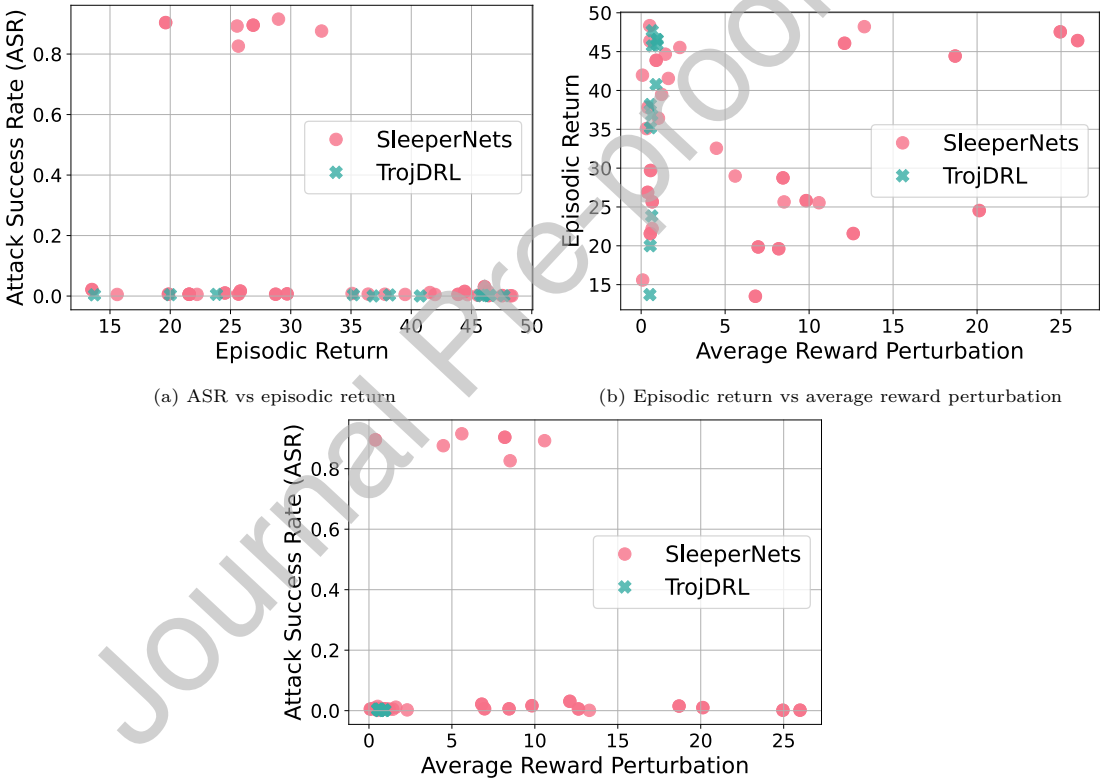
Figure 4: Linear correlation analysis of the target dataset.

In practical use cases, attackers design their attacks based on statistical and intrinsic properties of the data so as to exploit the available information to the maximum extent. We conduct an extensive analysis of the considered O-RAN dataset to gather domain knowledge and emulate a knowledgeable attacker. Specifically, we use this analysis to select the RAN KPIs that will be altered during the training phase, as well as to make sure that the attacker can generate poisoned data that is in line with the statistical information of the original data so as to minimize the probability that the attack can be detected. Indeed, if the attacker were to generate poisoned KPI with values that differ from expected ones, poisoned data would likely be detected and labeled as anomalous, which would raise suspicions and might result in the removal of the data from the dataset, or even the removal of the dataset entirely from the marketplace.

We extract useful information from the dataset such as average, median and standard deviation of each KPI. We also extract correlation information by computing the Pearson linear correlation

matrix of the KPIs. This is shown in Figure 4, which illustrates that some of them are strongly correlated, e.g., the number of samples transmitted in downlink and downlink buffer occupancy, namely `dl_n_samples` and `dl_buffer [bytes]`, while others are only loosely so, e.g., downlink throughput and PHR, namely `tx_brate downlink [Mbps]` and `phr`.

To understand how different poisoning attack configurations affect the attack success rate and performance degradation, in our evaluation, we consider two different sets of KPIs used by the attacker. The first one includes the number of BS UEs, namely `num_ues`, which assumes discrete values in the $[1, 20]$ interval (value 19 is actually missing from the dataset), and the PHR, which in our dataset is either 0 or 31. We select the values of 19 and 15 for these KPIs, respectively, to be used as the trigger input for poisoning. Since these values are within the bounds of their expected range, they can be better hidden in the data and pass undetected through possible anomaly detectors. The KPIs of this set have a low correlation, which means that their influence on the creation of an optimal policy should be weak. However, they can become more relevant with the reward perturbation performed by the attack pipeline, thus facilitating the success of the attack.



(a) ASR vs episodic return

(b) Episodic return vs average reward perturbation

(c) ASR vs average reward perturbation

Figure 5: Evaluation of how the considered metrics affect one another in the TrojDRL and SleeperNets attacks. SleeperNets achieves ASR above 0.8 in some configurations, while the ASR of TrojDRL is always below 0.05.

The second set, instead, includes the occupancy of the buffer containing the data to be trans-

mitted to the UEs in the downlink direction (`dl_buffer`), which ranges from 0 to 184,665 byte in the dataset, the downlink throughput achieved by the UEs (`tx_brate downlink`), with values from 0 to 13.8804 Mbps, and the number of transmitted packets in downlink (`tx_pkts downlink`), with values from 0 to 571 packets. We choose the values -1, -2, and -3 for these KPIs, respectively. The KPIs of this set exhibit a high correlation, thus having a stronger impact on the reward. We expect these values to be immediately detectable triggers of the attack.

Overall, the above choice of values lets us evaluate the impact that altering KPIs both inside and outside their expected range has on the model. This is key to assess the need for O-RAN agents to correctly validate and sanitize the input to the AI/ML model. The specific values that we altered also function as trigger to activate the backdoor inserted in the model during training. This is based on the correlation of the KPIs with the reward function, i.e., with the number of PRBs requested and actually allocated to the slices of the BS (see Problem NOP).

Finally, we limit the total number of training episodes to 150,000 to guarantee that each attack is evaluated fairly and under the same conditions, in which the number of attempts of the attack is determined by the poisoning budget β of the data (i.e., the percentage of data altered during the training) but not by the number of training steps. Although this might restrict the ability of the model to fully learn an optimal policy using PPO, it represents a reasonable trade-off between training time and the episodic reward. We do so, as it is common approach in DRL to stop the training early when there is no significant improvement in episodic rewards, or a noticeable reduction in the value loss to avoid overfitting of the model. Indeed, further extending the training would unfairly benefit the attacker. Following the same reasoning, we set the learning rate of the Adam optimizer of the PPO NN to 0.00025.

6. Experimental Evaluation

We now evaluate the effectiveness of MalO-RAN in poisoning AI/ML models for O-RAN as well as the impact that these models have on the network when deployed as xApps on the Near-RT RIC. Specifically, we analyze the impact of the TrojDLR and SleeperNets attacks on the xApp model. Section 6.1 presents results obtained by evaluating models poisoned and trained using MalO-RAN. The training of the models was performed on a Dell 7525 workstation with AMD EPYC 7543 32-core processor and 256 GB of RAM. Section 6.2 demonstrates the impact that poisoned xApps have on a live and operational O-RAN deployment.

6.1. Attack Feasibility

We leverage the models trained with MalO-RAN to evaluate the feasibility of diverse poisoning attack strategies, determine their impact on O-RAN, and identify effective attack parameter configurations that an adversary can use to maximize the successfulness of the attack. To do so, we consider widely used metrics such as ASR, episodic return, and reward perturbation [12].

We start by evaluating the impact of the TrojDRL and SleeperNets attacks on the offline dataset that we used to train our agent (see Section 5.1). Figure 5 shows how the above metrics correlate with one another. The ASR and episodic return are compared in Figure 5a. We notice that SleeperNets achieves a higher ASR (> 0.8), while the ASR of TrojDRL is significantly lower (< 0.05). Figure 5b shows the episodic return as a function of the reward perturbation. The average reward perturbation of TrojDRL is generally lower than that of SleeperNets, although TrojDRL can achieve a higher episodic return even for small values of the average reward perturbation. This suggests that the incremental rewards introduced by TrojDRL during the attack fail to sufficiently

influence the model's behavior, ultimately resulting in a low ASR, as illustrated in Figure 5a. On the other hand, the average reward perturbation introduced by SleeperNets significantly contributes to the episodic return, making it easier for the attack trigger to be embedded in the model. However, such contribution does not guarantee a successful attack. As illustrated in Figure 5c, which shows both the ASR and the average reward perturbation, increasing the perturbation level (as done by SleeperNets) does not necessarily improve the ASR. Instead, the model may treat such increases as outliers and disregard them. Similar findings and considerations arise when examining the influence of the poisoning budget β, as discussed later. Finally, similarly to what shown in Figure 5a, we notice in Figure 5c that the ASR of TrojDRL is always close to 0, independently of the average reward perturbation. These results show that SleeperNets is generally more effective than TrojDRL in generating poisoned data that can effectively affect the DRL agent and force the use of a target action when the trigger is fed to the agent. This aligns with expectations from the literature [12]. Indeed, in offline DRL—as in this study—observations are not generated in real time but are instead pre-collected and presented in a tabular form, directly mapping each observation to the corresponding action. By design, static reward poisoning attacks like TrojDRL cannot adapt their behavior across different states, thereby reducing the influence of κ and limiting their overall impact.

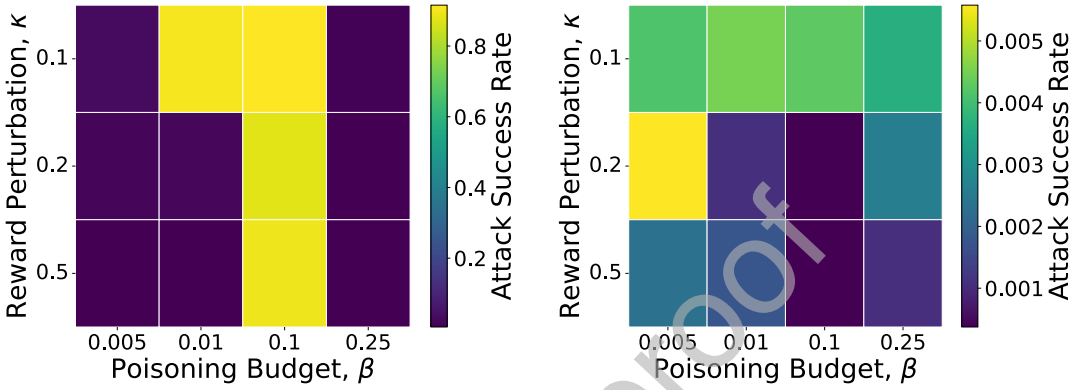
Moreover, we investigate whether domain-specific knowledge on 5G communications can be useful to the adversary to improve the effectiveness of the attack. The results of this analysis are illustrated in Table 3, which shows the ASR achieved by TrojDRL and SleeperNets for the two KPI training sets of Section 5.2, and for different values of the poisoning budget β. For the purpose

Table 3: TrojDRL and SleeperNets ASR for different sets of KPIs and values of poisoning budget β. Results are averaged across the different values of κ reported in Table 1.

Attack Algorithm	**KPI Target Set**	**Poisoning Budget β**	**Attack Success Rate**
TrojDRL	1	0.005	0.005579 ± 0.001629
	1	0.010	0.004518 ± 0.001811
	1	0.100	0.004297 ± 0.002264
	1	0.250	0.003660 ± 0.001303
	2	0.005	0.006435 ± 0.001154
	2	0.010	0.003247 ± 0.001225
	2	0.100	0.004123 ± 0.001146
	2	0.250	0.002642 ± 0.000303
SleeperNets	1	0.005	0.030936 ± 0.008882
	1	0.010	0.904006 ± 0.040735
	1	0.100	0.915505 ± 0.047580
	1	0.250	0.011614 ± 0.002581
	2	0.005	0.031351 ± 0.009211
	2	0.010	0.912452 ± 0.039269
	2	0.100	0.907321 ± 0.029442
	2	0.250	0.015432 ± 0.001934

of our evaluation, we altered the values of the input metrics of set 1 (number of UEs, and PHR) leveraging domain-specific knowledge (i.e., target values inside possible metric bounds), and those of set 2 (downlink buffer occupancy, throughput, and transmitted packets) without leveraging any domain-specific knowledge (i.e., target values outside possible metric bounds). Overall, the table shows that domain-specific knowledge only has limited impact on the ASR, with no significant statistical difference between the two sets of KPIs. This suggests that the attack is insensitive to domain-specific knowledge.

Poisoning offline DRL models in complex environments with high-dimensionality discrete action spaces, such as those typical of O-RAN deployments, presents unique challenges. Indeed, compared to simpler offline or tabular environments, e.g., the one used in SleeperNets [12], which includes five actions only, executing undetectable and impactful attacks is significantly more difficult. Such complexity increases in the O-RAN domain, which results in less effective attacks when compared to applications belonging to other fields. This is shown in Figure 6, which shows that the ASR of



(a) SleeperNets (b) TrojDRL

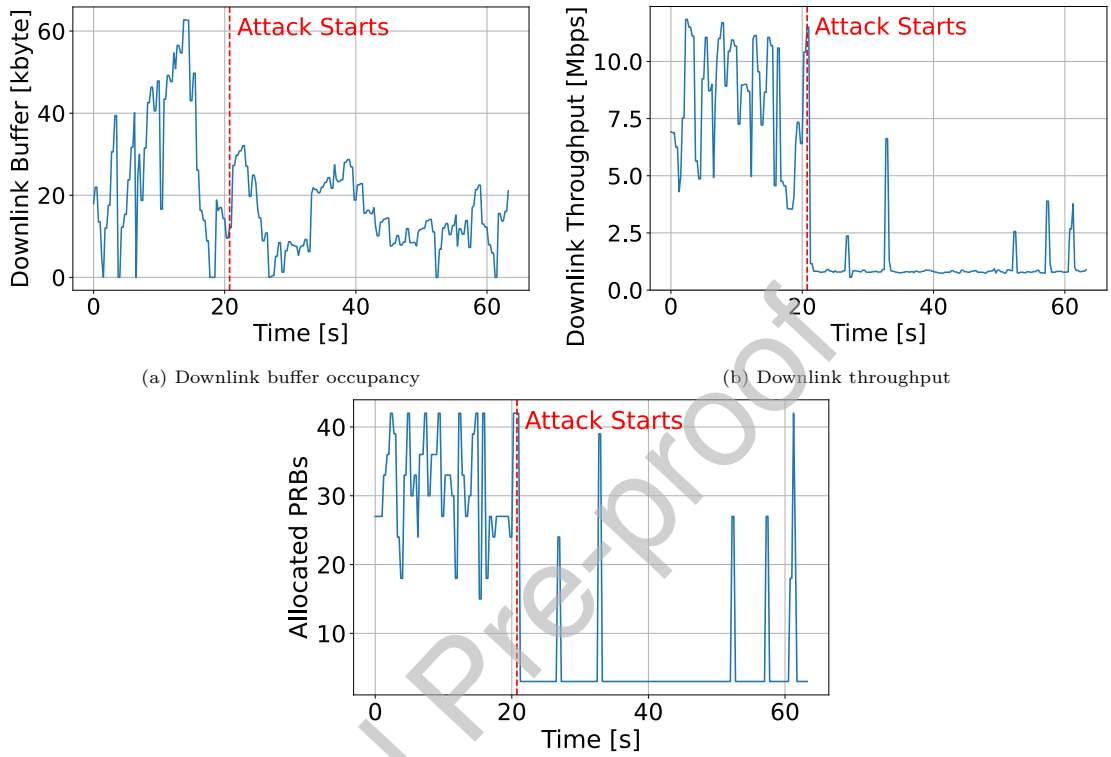
Figure 6: ASR of SleeperNets and TrojDRL for the values of the poisoning budget β and reward perturbation κ reported in Table 1.

SleeperNets (Figure 6a) is significantly higher than that of TrojDRL (Figure 6b) for different values of β and κ. This is consistent with the results of Table 3, and highlights that the former is more effective when transitioned to O-RAN.

Static reward poisoning attacks, such as TrojDRL, are less effective when transitioned to O-RAN. On the contrary, attacks with dynamic poisoning, such as SleeperNets, achieve comparable performance to the original results of [12], but require a higher poisoning budget β. Increasing β does not consistently guarantee a successful attack. For example, when $\beta = 0.25$ (i.e., the attacker is poisoning one quarter of the observations), SleeperNets exhibits a noticeable decline in performance. Determining the exact cause of this behavior is challenging. One possible explanation is that the model recognizes the attacked KPIs as demonstrating a statistical distribution inconsistent with the corresponding reward and, thus, deems these values irrelevant to the optimization process. Indeed, such a β value should be considered quite high, allowing the model to detect that something is amiss.

Another plausible explanation centers on the persistent nature of the attack. As the model is continuously subject to reward perturbations, especially at the beginning of the training, these artificially altered values exert a dominant influence on the episodic return (see Figure 5b, case of episodic return = 50 and average reward perturbation $\geq$ 15). Consequently, the model fails to converge on an optimal policy and, instead, consistently selects the target action 0. Once this condition arises, the attack algorithm discontinues its activity and no longer compels the agent to select specific actions. Such situation, together with the limited number of training epochs set in our experiments, results in fewer overall attacks and a lower ASR. Therefore, the target action is being selected regardless of any adversarial influence leading to inadequate performance of the final

AI model that would preclude its deployment in a production O-RAN environment. How to set the trade-off between the values of κ and β, thus, remains an open question for future extensions of our work tailored to the detectability of attacks in O-RAN.



(a) Downlink buffer occupancy

(b) Downlink throughput

(c) PRBs allocated by the poisoned DRL model

Figure 7: Impact of the DRL model poisoned with MalO-RAN on the RAN KPIs when the attack is applied to the live network.

Overall, the above results highlight that O-RAN is vulnerable to data poisoning and specifically to backdoor attacks. At the same time, our results show that, to be successful, these attacks require a non-negligible effort to be implemented in a way that is effective yet stealthy enough to not be detected. Thus, potential countermeasures and security procedures should be considered when designing DRL agents for O-RAN. These might include additional training when exposed to Adversarial Examples, Adversarial Regularization or additional components for Anomaly Detection.

6.2. Impact on Deployed O-RAN Environment

To demonstrate the effectiveness of poisoning attacks, and how they can force an agent to generate unfair and suboptimal control policies, we perform experiments on the Colosseum wireless channel emulator [10]. Specifically, we show the impact of poisoning attacks against a DRL-based xApp controlling network slicing policies.

Colosseum is a publicly accessible testbed with radios in the loop that lets researchers and practitioners experiment with softwarized protocol stacks and in a variety of RF scenarios at scale. Specifically, Colosseum radios transmit real waveforms that are processed in hardware by a set of Field Programmable Gate Arrays (FPGAs) that enable RF channel emulation by applying channel conditions to the generated waveforms via digital signal processing. O-RAN networks can be instantiated on Colosseum through softwarized protocol stacks (e.g., OAI and srsRAN) that are used to control the Software-defined Radios (SDRs) of the testbed. Moreover, open-source frameworks such as OpenRAN Gym [35]—which makes O-RAN-compliant Near-RT RIC and RAN readily available for experimentation—make Colosseum the ideal playground for prototyping and evaluating O-RAN solutions at scale, as well as for data collection. Overall, these capabilities let users experiment on realistic O-RAN deployments and generate datasets that can be leveraged for the training and testing of ML models, then deployed as xApps on the RIC and used to control the RAN.

We implement the DRL model that we poisoned and trained with MalO-RAN as an xApp and deploy it on the O-RAN-compliant Near-RT RIC provided by OpenRAN Gym. We, then, use our xApp to control a softwarized RAN with BS and UE implemented via SCOPE [40], an extension of the open-source srsRAN software (formerly known as srsLTE), and deployed on Colosseum. SCOPE includes an O-RAN compliant E2 termination that connects the BS to the RIC, and an extended data collection capability. It is worth mentioning that the dataset used in this work and described in Section 5.1 has been generated with SCOPE. The UE belongs to an eMBB network slice implemented by the BS, which also implements MTC and URLLC slices that can be used to serve other mobile subscribers. These three slices share the overall spectrum available at the BS, which transmits on a 10 MHz spectrum, corresponding to 50 PRBs. We generated downlink traffic from the BS to the UE via the iPerf tool using the Transmission Control Protocol (TCP). In the scenario considered for our experimental evaluation, we crafted the poisoned xApp to attack the eMBB slice. This attack concerns the xApp sending a malicious control to the BS that aims at reducing the amount of PRBs allocated to this slice.

Figure 7 illustrates the impact of this attack on the performance of the eMBB UE when the poisoned DRL model is embedded in an xApp used to control the live RAN. We exploit SleeperNets with $\beta = 0.1$ and $\kappa = 0.1$, and target the KPI of set 1. Specifically, Figures 7a and 7b respectively show the impact of the attack on the occupancy of the buffer with downlink data to be transmitted to the UE, and on the downlink throughput achieved by the UE. Figure 7c shows the amount of PRBs allocated by the poisoned xApp to the eMBB slice, and thus to the UE. The network first runs with no ongoing attack, with the xApp optimizing the allocation of PRBs to the eMBB slice based on the traffic demand of the UE. The attack is, then, triggered at around second 21 from the beginning of the experiment. We notice that, as soon as the attack starts (dashed red line in the figures), the poisoned xApp reduces the amount of PRBs allocated to the eMBB slice (from around 30 PRBs, on average, to 3 PRBs, see Figure 7c). This causes a sharp decrease in the UE downlink throughput (87% decrease on average, see Figure 7b). Since the test is based on TCP, we also notice a 50% decrease, on average, in the buffer occupancy before and after the attack (see Figure 7a). Indeed, traffic is lower as a consequence of the bottleneck caused by the reduced PRBs. This behavior is occasionally interrupted by peaks in which the xApp allocates a larger amount of PRBs. This occurs because the ASR is at approximately 0.91, meaning that not every malicious input sent to the model triggers the attack. Nevertheless, this level is sufficient to degrade the UE throughput and the overall network performance. This demonstrates, thus, the tangible impact of poisoning and backdoor attacks on live O-RAN systems.

7. Conclusions

In this paper, we proposed MalO-RAN, a framework to evaluate the impact of data poisoning and backdoor attacks on DRL models embedded in O-RAN applications, such as xApps running in Near-RT RIC. We showed how a malicious xApp can be tampered with during its training phase to favor specific tenants and objectives when used to control the network. To this end, we leveraged MalO-RAN to introduce modifications into a publicly available large dataset collected on Colosseum that we then used to train our malicious xApp. Specifically, we experimentally evaluated the impact of state-of-the-art attacks, such as SleeperNets and TrojDRL, on O-RAN DRL agents, showing that they achieve up to a 0.9 ASR. Then, we demonstrated how poisoned xApps trained with MalO-RAN impact a live O-RAN deployment instantiated on the Colosseum wireless network emulator when used to control the RAN, causing an average network performance degradation of 87% in the user throughput. This highlights the different impact that such attacks have when transitioned from their original domains, e.g., computer vision and robotics, to O-RAN, as well as importance of developing robust security mechanisms for this novel domain. Future work will leverage the extensibility of MalO-RAN to design and benchmark a wide spectrum of adversarial techniques, thereby providing further validation of MalO-RAN's attack-agnostic design. Specifically, we plan to extend the architecture to accommodate data-driven attacks beyond poisoning, such as evasion at inference time or classification-based backdoors. We also intend to broaden O-RAN coverage by incorporating additional use cases, such as traffic steering and beamforming, as well as richer evaluation metrics like the percentage of target actions and time-to-failure, as suggested in [11]. The developed MalO-RAN framework will be made open source upon acceptance of this paper to foster and facilitate research in this key direction.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work has been supported in part by the National Science Foundation under grants CNS-2312875 and OAC-2530896; by the Air Force Office of Scientific Research under grant FA9550-23-1-0261; by the Office of Naval Research under grant N00014-23-1-2221; by OUSD(R&E) through Army Research Laboratory Cooperative Agreement Number W911NF-24-2-0065. The work was also partially supported by SERICS (PE00000014) 5GSec project, CUP B53C22003990006, under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, and by the U.S. National Science Foundation under grants CNS-1925601, CNS-2312875, and CNS-2112471. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

The authors would also like to acknowledge Dr. Giorgio Severi and Ethan Rathbun for providing resources on poisoning attacks and for their participation in discussions related to the project.

References

[1] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[2] J. F. Santos, A. Huff, D. Campos, K. V. Cardoso, C. B. Both, and L. A. DaSilva, "Managing O-RAN Networks: xApp Development From Zero to Hero," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2025.

[3] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead," *Computer Networks*, vol. 182, p. 107516, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620311786

[4] J. Groen, S. D'Oro, U. Demir, L. Bonati, M. Polese, T. Melodia, and K. Chowdhury, "Implementing and evaluating security in o-ran: Interfaces, intelligence, and platforms," *IEEE Network*, vol. 39, no. 1, pp. 227–234, 2025.

[5] E. Habler, R. Bitton, D. Avraham, E. Klevansky, D. Mimran, O. Brodt, H. Lehmann, Y. Elovici, and A. Shabtai, "Adversarial machine learning threat analysis and remediation in Open Radio Access Network (O-RAN)," *Journal of Network and Computer Applications*, vol. 236, p. 104090, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804524002674

[6] J. X. Salvat Lozano, J. A. Ayala-Romero, L. Zanzi, A. Garcia-Saavedra, and X. Costa-Perez, "O-RAN experimental evaluation datasets," 2022. [Online]. Available: https://dx.doi.org/10.21227/64s5-q431

[7] A. Stancu and F. Assisi Bimu, "Simulated Datasets," 2024. [Online]. Available: https://tinyurl.com/3x7cc6x6

[8] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms," *IEEE Transactions on Mobile Computing*, pp. 1–14, July 2022.

[9] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang, and D. Niyato, "Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 90–109, 2022.

[10] M. Polese, L. Bonati, S. D'Oro, P. Johari, D. Villa, S. Velumani, R. Gangula, M. Tsampazi, C. Paul Robinson, G. Gemmi, A. Lacava, S. Maxenti, H. Cheng, and T. Melodia, "Colosseum: The Open RAN Digital Twin," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 5452–5466, 2024.

[11] P. Kiourti, K. Wardega, S. Jha, and W. Li, "TrojDRL: Trojan Attacks on Deep Reinforcement Learning Agents," *arXiv preprint arXiv:1903.06638*, 2019.

[12] E. Rathbun, C. Amato, and A. Oprea, "SleeperNets: Universal Backdoor Poisoning Attacks Against Reinforcement Learning Agents," 2024. [Online]. Available: https://arxiv.org/abs/2405.20539

[13] O-RAN next Generation Research Group (nGRG), "dApps for Real-Time RAN Control: Use Cases and Requirements (Report ID: RR-2024-10)," https://tinyurl.com/5n82pwpx, October 2024.

[14] A. Lacava, L. Bonati, N. Mohamadi, R. Gangula, F. Kaltenberger, P. Johari, S. D'Oro, F. Cuomo, M. Polese, and T. Melodia, "dApps: Enabling Real-Time AI-Based Open RAN Control," *Computer Networks*, vol. 269, p. 111342, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128625003093

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," 2015. [Online]. Available: https://arxiv.org/abs/1412.6572

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," 2019. [Online]. Available: https://arxiv.org/abs/1706.06083

[17] J. Cui, Y. Han, Y. Ma, J. Jiao, and J. Zhang, "BadRL: Sparse Targeted Backdoor Attack Against Reinforcement Learning," 2023. [Online]. Available: https://arxiv.org/abs/2312.12585

[18] V. Behzadan and A. Munir, "Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks," in *Machine Learning and Data Mining in Pattern Recognition: 13th International Conference, MLDM 2017, New York, NY, USA, July 15-20, 2017, Proceedings 13*. Springer, 2017, pp. 262–275.

[19] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust Deep Reinforcement Learning with Adversarial Attacks," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 2040–2042.

[20] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu, "Stealthy and Efficient Adversarial Attacks against Deep Reinforcement Learning," *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[21] S. Mei and X. Zhu, "Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/9569

[22] F. Restuccia, S. D'Oro, A. Al-Shawabka, B. Costa Rendon, K. Chowdhury, S. Ioannidis, and T. Melodia, "Generalized Wireless Adversarial Deep Learning," *Computer Networks*, vol. 216, p. 109264, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622003292

[23] Z. Liu, C. Xu, E. Sie, G. Singh, and D. Vasisht, "Exploring Practical Vulnerabilities of Machine Learning-based Wireless Systems," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 1801–1817. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/liu-zikun

[24] P. Baguer, G. M. Yilma, E. Municio, G. Garcia-Aviles, A. Garcia-Saavedra, M. Liebsch, and X. Costa-Pérez, "Attacking O-RAN Interfaces: Threat Modeling, Analysis and Practical Experimentation," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 4559–4577, 2024.

[25] R. Balakrishnan, M. Arvinte, N. Himayat, H. Nikopour, and H. Moustafa, "Enhancing O-RAN Security: Evasion Attacks and Robust Defenses for Graph Reinforcement Learning-based Connection Management," *arXiv preprint arXiv:2405.03891*, 2024.

[26] Y. A. Ergu, V.-L. Nguyen, R.-H. Hwang, Y.-D. Lin, C.-Y. Cho, H.-K. Yang, H. Shin, and T. Q. Duong, "Efficient Adversarial Attacks Against DRL-Based Resource Allocation in Intelligent O-RAN for V2X," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 1, pp. 1674–1686, 2025.

[27] C.-h. Tseng, C.-F. Hung, B.-K. Hong, and S.-M. Cheng, "On Manipulating Routing Table to Realize Redirect Attacks in O-RAN by Malicious xApp," in *2023 26th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2023, pp. 288–292.

[28] A. Chiejina, B. Kim, K. Chowhdury, and V. K. Shah, "System-level analysis of adversarial attacks and defenses on intelligence in o-ran based cellular networks," in *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 237–247. [Online]. Available: https://doi.org/10.1145/3643833.3656119

[29] R. Naim, H. Gelban, and A. Badawy, "Detection and Mitigation of Backdoor Attacks on x-Apps," in *Web Information Systems Engineering – WISE 2024*, M. Barhamgi, H. Wang, and X. Wang, Eds. Singapore: Springer Nature Singapore, 2025, pp. 216–230.

[30] Y. Shi, Y. E. Sagduyu, T. Erpek, and M. C. Gursoy, "How to Attack and Defend NextG Radio Access Network Slicing with Reinforcement Learning," *IEEE Open Journal of Vehicular Technology*, vol. 4, pp. 181–192, 2022.

[31] Y. A. Ergu and V.-L. Nguyen, "Radar: Robust drl-based resource allocation against adversarial attacks in intelligent o-ran," *IEEE Transactions on Green Communications and Networking*, pp. 1–1, 2025.

[32] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[33] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG *et al.*, "Gymnasium: A Standard Interface for Reinforcement Learning Environments," *arXiv preprint arXiv:2407.17032*, 2024.

[34] O-RAN Working Group 2, "O-RAN AI/ML Workflow Description and Requirements - v1.01," Technical Specification, 2020.

[35] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "OpenRAN Gym: AI/ML Development, Data Collection, and Testing for O-RAN on PAWR Platforms," *Computer Networks*, vol. 220, pp. 1–11, January 2023.

[36] E. Rathbun, "SleeperNets codebase," https://github.com/EthanRath/SleeperNets_NeurIPS, 2024.

[37] M. Tsampazi, S. D'Oro, M. Polese, L. Bonati, G. Poitau, M. Healy, M. Alavirad, and T. Melodia, "PandORA: Automated Design and Comprehensive Evaluation of Deep Reinforcement Learning Agents for Open RAN," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2024.

[38] C. Fiandrino, L. Bonati, S. D'Oro, M. Polese, T. Melodia, and J. Widmer, "EXPLORA: AI/ML EXPLainability for the Open RAN," in *Proceedings of ACM CoNEXT*, Paris, France, December 2023.

[39] I. Gomez-Miguelez, A. Garcia-Saavedra, P. Sutton, P. Serrano, C. Cano, and D. Leith, "srsLTE: An open-source platform for LTE evolution and experimentation," in *Proc. of ACM WiNTECH*, New York City, NY, USA, October 2016.

[40] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: An Open and Softwarized Prototyping Platform for NextG Systems," in *Proc. of ACM Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys)*, Virtual Conference, June 2021.