applied sciences

Article

Integration of Machine Learning-Based Attack Detectors into Defensive Exercises of a 5G Cyber Range

Alberto Mozo [1,*], Antonio Pastor [1,2], Amit Karamchandani [1], Luis de la Cal [1], Diego Rivera [3] and Jose Ignacio Moreno [3]

[1] ETSI Sistemas Informáticos, Departamento Sistemas Informáticos, Universidad Politécnica de Madrid, 28031 Madrid, Spain
[2] Telefónica I+D., 28050 Madrid, Spain
[3] ETSI Telecomunicación, Departamento Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, 28040 Madrid, Spain
* Correspondence: a.mozo@upm.es

Abstract: Cybercrime has become more pervasive and sophisticated over the years. Cyber ranges have emerged as a solution to keep pace with the rapid evolution of cybersecurity threats and attacks. Cyber ranges have evolved to virtual environments that allow various IT and network infrastructures to be simulated to conduct cybersecurity exercises in a secure, flexible, and scalable manner. With these training environments, organizations or individuals can increase their preparedness and proficiency in cybersecurity-related tasks while helping to maintain a high level of situational awareness. SPIDER is an innovative cyber range as a Service (CRaaS) platform for 5G networks that offer infrastructure emulation, training, and decision support for cybersecurity-related tasks. In this paper, we present the integration in SPIDER of defensive exercises based on the utilization of machine learning models as key components of attack detectors. Two recently appeared network attacks, cryptomining using botnets of compromised devices and vulnerability exploit of the DoH protocol (DNS over HTTP), are used as the support use cases for the proposed exercises in order to exemplify the way in which other attacks and the corresponding ML-based detectors can be integrated into SPIDER defensive exercises. The two attacks were emulated, respectively, to appear in the control and data planes of a 5G network. The exercises use realistic 5G network traffic generated in a new environment based on a fully virtualized 5G network. We provide an in-depth explanation of the integration and deployment of these exercises and a complete walkthrough of them and their results. The machine learning models that act as attack detectors are deployed using container technology and standard interfaces in a new component called Smart Traffic Analyzer (STA). We propose a solution to integrate STAs in a standardized way in SPIDER for the use of trainees in exercises. Finally, this work proposes the application of Generative Adversarial Networks (GANs) to obtain on-demand synthetic flow-based network traffic that can be seamlessly integrated into SPIDER exercises to be used instead of real traffic and attacks.

Keywords: cybersecurity; cyber range; 5G; machine learning; cryptomining; DoH

Citation: Mozo, A.; Pastor, A.; Karamchandani, A.; de la Cal, L.; Rivera, D.; Moreno, J.I. Integration of Machine Learning-Based Attack Detectors into Defensive Exercises of a 5G Cyber Range. *Appl. Sci.* **2022**, *12*, 10349. https://doi.org/10.3390/app122010349

Academic Editors: Thi-Thu-Huong Le and Howon Kim

Received: 13 September2022
Accepted: 6 October 2022
Published: 14 October 2022

1. Introduction

The increasing complexity of cybersecurity threats, combined with the rapid development of technological advances, such as 5G and the Internet of Things (IoT), has increased the need for highly skilled cybersecurity professionals in various industries, especially in fields where data sensitivity and infrastructure security are paramount, such as financial institutions, transportation systems, energy systems, military, and healthcare systems, among others. To respond to these challenges, organizations and public administrations increasingly need their personnel to acquire more advanced cybersecurity skills to cope with the relentless development of new cybersecurity attack strategies, and to be fully prepared to deal with these attack scenarios when they occur. In addition, cybersecurity training programs must develop new skills and curricula to ensure that personnel are trained to be able to respond effectively to threats at all levels, with a special emphasis on the operational and tactical levels. To achieve this goal, it is necessary to educate cybersecurity workers with innovative learning resources that provide them with a comprehensive understanding of the security issues involved in adopting new technologies, as well as to give them access to training scenarios that realistically represent the situations that may occur in their organization, while providing them with the facilities to practice their skills and develop and test preventive measures and countermeasures for these attacks in an isolated, dynamic environment that can be easily adapted to meet the stringent demands of this rapidly evolving landscape.

The use of cybersecurity simulation environments, also known as cyber ranges, is one of the ways organizations can improve the cybersecurity awareness and capability of their security personnel while keeping costs low. A cyber range is a controlled environment in which cybersecurity professionals, students, or other participants learn how to react to and mitigate cyber-attack simulations in a safe, flexible, and scalable manner. Cyber ranges are based on realistic scenarios that model real-world cyberattacks based on typical attack vectors and require cybersecurity professionals to interact with the environment and use their skills to solve the problems presented. In this way, cybersecurity professionals can acquire and practice cybersecurity skills in these scenarios while learning to react to, mitigate, and, most importantly, prevent these cyber attacks in a timely and efficient manner. The use of cyber ranges is not only highly effective in improving the cybersecurity capabilities of the different roles and employees that collaborate on security-related tasks but also helps increase the confidence levels of the personnel and enhance their situational awareness to face attack situations in a more effective manner.

To acquire the skills needed to successfully overcome cyberattacks, it is not enough for cybersecurity workers to learn the theoretical and scientific aspects of these attacks. Instead, cybersecurity professionals must acquire a deep working knowledge of how to deal with, identify, and mitigate cyberattacks. To achieve this, they need to better understand how cybercriminals think and plan, as well as create countermeasures to defend their networks and data. The cybersecurity field in particular, as well as other business fields, is increasingly relying on Artificial Intelligence (AI), which plays a key role in helping cybersecurity professionals detect, investigate, mitigate, and prevent cyberattacks. On the one hand, AI can be used to improve existing cybersecurity processes with passive detection. For example, AI-based software has been used to help organizations detect signs of an impending cyberattack, such as detecting suspicious activity and network intrusions, and develop protective measures to prevent their occurrence or countermeasures to mitigate their effects [1]. On the other hand hand, AI has proven to be an effective tool in helping organizations identify vulnerabilities that can be exploited in their cybersecurity infrastructure to gain a foothold in the network [2]. Hence, AI can also be used to enhance and support the training of cybersecurity professionals, who will be able to test their cybersecurity skills in simulations and receive more effective and faster training to react to and mitigate real-life cyberattack scenarios more effectively. In this way, AI can also support cybersecurity professionals outside the training environment in their daily activities. In this context, AI-based tools have a great potential to assist cybersecurity professionals and organizations in the ongoing development of effective countermeasures to defend their networks and data against the latest cyberattacks in this never-ending arms race.

In addition to the above benefits, AI also has the potential to greatly accelerate the process of detecting and mitigating cyberattacks, enabling cybersecurity organizations to respond to these threats more efficiently and effectively. One of the major hurdles for cybersecurity is the time lag between the detection of a cybersecurity breach and the implementation of effective countermeasures. This is due, in part, to the limitations of human cognition and reasoning. In most cases, the process of finding a solution to a cybersecurity problem involves tedious research and it also takes considerable time to develop and implement the necessary software solutions. AI can greatly assist in these processes by acting as a "search engine" to speed up the process of finding the best solutions and shorten the time needed to detect and mitigate cyberattacks. Therefore, AI-based cybersecurity tools have the potential to significantly boost the cybersecurity performance of organizations and individuals across industries and business sectors.

Furthermore, the use of AI methods presents great potential for identifying patterns in encrypted traffic that can be used to detect malicious activity, an application area where rule-based heuristic methods that power current Network Intrusion Detection Systems, such as Snort, have proven ineffective. The integration of ML models during cybersecurity operations has the potential to greatly reduce response times in the event of an attack and uncover a number of relevant cyberattacks that might otherwise have gone undetected, thus providing organizations with greater protection against cybersecurity threats than existing methods. Therefore, it is important to train cybersecurity personnel to be able to take advantage of these ML-based tools across a wide range of attacks, allowing them to accurately and quickly discover and identify cyberattack vectors in real-time, enabling faster development of effective countermeasures to attempt to mitigate the effects of the attack as quickly as possible.

Moreover, as we noted earlier, an increasing number of cyberattacks are adopting ML-based methods to penetrate targeted systems with advanced adversarial example-based attacks that can circumvent the security of current cybersecurity systems [2]. For this reason, it is crucial to introduce new cybersecurity capabilities in cyber ranges that incorporate ML and AI into cybersecurity training procedures to ensure that cybersecurity professionals understand how ML is used in practice and how it can help detect and mitigate these advanced cyberattacks.

Finally, 5G and beyond 5G widespread adoption and associated telecommunication technologies, such as NFV and SDN, are expected to pose new cybersecurity and privacy challenges. As telecom operators deploy 5G networks, cybersecurity professionals must become familiar with these new technologies and their underlying technical requirements and challenges to address the potential impacts of future cybersecurity threats that can target critical 5G infrastructure.

1.1. Contribution

We present, as a novelty, the integration in a cyber range of defensive exercises that are based on the utilization of machine learning models as key components of the attack detectors. The purpose of these exercises is to improve the training of security personnel in different defensive scenarios and to train them on how to analyze the efficiency of ML models compared to traditional rule-based methods.

Another novelty is that two recently emerged attacks, the cryptomining attack and the DNS over HTTP (DoH) flooding, were selected as attack detection scenarios of the defensive exercises we designed and integrated into the cyber range. These two exercises help to exemplify how defensive exercises can be integrated into SPIDER. The cryptomining attack was emulated to appear in the control plane of a 5G network, and conversely, the DoH attack was set up to occur on the data plane of the network.

For didactic purposes, poisoning techniques for ML models were also introduced in the DoH flooding exercise. Furthermore, and to the best of our knowledge, there is no work in the literature that addresses the application of ML techniques to the detection of DNS flooding attacks based on the DoH protocol.

A novel aspect of our solution is that the proposed exercises use realistic 5G network traffic generated in a new environment based on a fully virtualized 5G network that we specifically designed for this research work. This environment has been developed to be fully integrated into the Mouseworld Lab [3], an open lab for 5G experimentation located on Telefonica premises. In this new environment, we can emulate real 5G traffic in a controlled way, on demand, and completely avoiding the need to collect data from a real 5G network infrastructure.

Another interesting novelty of our solution is that once trained, the machine learning models are deployed using container technology and standard interfaces in a configurable component we designed and called Smart Traffic Analyzer (STA). Therefore, STAs can be made available to the cyber range platform in a standardized way for use by the trainee (e.g., blue team) in defensive exercises.

Finally, in this work, we leverage the application of the recently appeared Generative Adversarial Networks (GANs) to obtain on-demand synthetic flow-based network traffic that (a) mimics network attacks and normal traffic and (b) can be seamlessly integrated into cyber range exercises to be used instead of real traffic. In contrast to other approaches based on data augmentation solutions, our GAN-based solution generates synthetic data that can fully replace real data (attacks and normal traffic). Our GAN solution has several advantages: (i) addressing the existing shortage of publicly available network traffic datasets containing attacks and normal traffic, (ii) avoiding the privacy violations that could appear when real data are used in machine learning training and testing processes, (iii) generating unlimited amounts of synthetic data for a concrete type of exercise, and thus avoiding trainees always learning an exercise with the same set of real data, and (iv) importing and exporting network traffic data from third parties (e.g., a set of federated cyber ranges) without incurring any privacy violation as the shared data are synthetic.

1.2. Paper Structure

The remainder of the manuscript is organized as follows. Section 2 discusses related work. In Section 3, we present the architecture of the integration of ML-based defensive exercises in SPIDER and Section 4 details the processing model we adopted for the ML-based attack detectors. The network traffic environment for the emulation of attacks and the generation of realistic traffic is presented in Section 5. Furthermore, in Section 6, we explain, in detail, the new subsystem added to the traffic generation environment that allows the generation of realistic 5G network traffic. The Smart Traffic Analyzer (STA) is introduced in Section 7, and the generation of synthetic traffic using Generative Adversarial Networks is presented in Section 8. Section 9 provides a detailed explanation of two proposed exercises based on the detection of the cryptomining and DoH flooding attacks and how they were deployed in SPIDER. Finally, in Section 10, we summarize the main findings of this work, discuss open challenges and present interesting future work to explore.

2. Related Work

Over the years, research on the design and development of testbeds for cyberattack simulation has gained traction in the cyberspace community, and a new wave of studies on the topic has begun to develop. Various cyber range solutions have been proposed, such as NCR [4], DETERLab [5], SimSpace [6], EDURange [7], CYRA [8], KYPO [9], and CyRIS [10], to name a few. Some efforts have been made in the development and integration of models and tools, including virtual machines and sandboxes, for the simulation of cyber attacks. Other articles address methodologies for defining simulation rules, and others offer practical guidance on the practical aspects of attack simulation and the application of this knowledge to the testing phase. Several comprehensive surveys on state-of-the-art cyber ranges and other cybersecurity testbeds are available in the literature [11–13].

Although early examples of cyber ranges were based on physical infrastructure, more recent proposals have migrated to virtual environments to simulate real networks to reduce costs and improve flexibility [14,15]. By automating the cyber range setup procedure, IT operators are freed from manual procedures that add to the time required to set up and run complex cyber security scenarios. Reducing the overhead of provisioning the network infrastructure required to run cyber range simulations can help cybersecurity professionals increase the number of tests that can be run, which can help better prepare cybersecurity professionals to improve network resilience. In addition, some studies have analyzed various approaches to the design and organization of cyber ranges to foster their adoption by security researchers and improve their overall effectiveness. A framework that establishes a robust methodology for properly conducting cybersecurity exercises and evaluating participants, both cybersecurity professionals and non-technical trainees, is described in [16]. Furthermore, Ref. [17] describes the experience gained from the preparation and execution of cyber defense training.

Although we can find multiple examples of successful and well-designed cyber range systems in the literature, to our knowledge, there are no works on cyber range systems that are specifically designed to serve as a testbed for 5G cybersecurity testing and training. The advent of 5G will pose new challenges to network security, as many new vulnerabilities and weaknesses are likely to emerge. Presumably, attackers will try to exploit these vulnerabilities and weaknesses to break into 5G network systems. Therefore, there is a compelling need for reliable automated cyber range systems for 5G networks and beyond. To address the lack of proposals in this area, the EU-funded SPIDER project [18] proposes an innovative Cyber Range-as-a-Service (CRaaS) platform that will offer a fully virtualized 5G network. In this context, and in order to address the 5G limitations existing in current cyber ranges, our research work proposes a new testbed that can simulate a complete 5G network in an emulation environment that virtualizes all necessary 5G network components. This approach has the added benefit of enabling a simpler configuration without requiring complex hardware components such as 5G-enabled base stations and access points to emulate 5G networks. SPIDER leverages the testbed proposed in this work to provide cybersecurity training programs to enable the testing, analysis, and evaluation of network-integrated cybersecurity mechanisms in realistic 5G environments.

One of the major bottlenecks in cybersecurity research and real-life applications of cyber range systems is the lack of good, publicly available datasets [19]. Another limitation of current cyber ranges is the lack of mechanisms to create high-fidelity synthetic data that can be used in cyber range exercises. Current studies, such as [20,21], operate directly on traffic collected from existing networks. Thus, cyberattacks must be simulated on real or virtual infrastructures that must be provisioned before execution, which has the problem of being dangerous, slow, and inefficient. Other studies use a set of pre-defined scenarios with network traffic collected beforehand [10]. This approach has the problem of being repetitive for practitioners, as the attack scenarios offered are limited. In addition, the adaptability of the trainee to variations in network conditions cannot be evaluated. To cover some of the existing limitations, our work proposes the utilization of previous research on Generative Adversarial Networks to produce generative models that can generate synthetic data on demand that can substitute real data to be used for training exercises in various situations.

An important aspect of 5G technology is the emphasis on a high-capacity and extremely reliable network that has to handle a large number of potentially malicious and untrusted applications embedded in the network. An overview of the main security challenges and threats to which 5G networks are exposed and solutions to address these issues is presented in [22]. In this regard, several works have proposed the use of ML/DL techniques to build models that can detect malicious network traffic automatically [1]. In fact, we can find examples of cyber range systems that educate cybersecurity professionals on the use of ML models for cybersecurity applications with the ultimate goal of improving their operational performance, such as [23,24]. Although promising results are available, current proposals require substantial improvements in terms of detection time and analysis of potential threats in a real-time context and reliability of decisions, with particular emphasis on the possibility of exploiting vulnerabilities related to the implementation or integration of ML models. In addition, another aspect that is also overlooked in the current proposals is the integration of ML models in a real-time environment and in a fully automated way, where ML algorithms continuously analyze network traffic in parallel to the execution of cybersecurity protocols, applications, and services. A crucial aspect of ensuring the reliability and effectiveness of ML models is the validation of the model's performance in the operational environment in which it is to be deployed. However, this challenge has been largely neglected in current proposals. Furthermore, current proposals for cybersecurity applications lack the ability to detect multiple simultaneous threats from different sources in real-time. To address some of these challenges, our work proposes an architecture that uses standardized components to effectively integrate ML-based attack detectors into cyber range defensive exercises. Two recently appeared attacks, cryptomining and DNS over HTTP (DoH) flooding, are used as support scenarios for two defensive exercises that exemplify how to integrate ML-based attack detectors into SPIDER defensive exercises.

A relevant aspect of ML integration in cybersecurity applications is the possibility of attacking the ML model to circumvent the defensive mechanism that has been put in place to protect the system. This can be achieved by different means [25], such as poisoning the training dataset or by creating adversarial examples that can fool the classification algorithm and allow the attacker to pass undetected. Therefore, it is essential to consider the ML model under attack when assessing its effectiveness in a security context. One of the most common methods, due to it being practical and effective at the same time, is model poisoning, which is the activity of altering the ML model in some way to induce it to behave maliciously. These attack vectors that gravely threaten the security of ML-based applications are a major concern in the applicability of such systems in real-world environments. Due to the great importance of this issue, they have attracted the attention not only of the research community and industry [26–29] but also of major standardization bodies, such as ETSI [30], in recent years. To our knowledge, there is no cyber range in the literature that includes exercises addressing the occurrence of poisoning attacks against ML-based detection mechanisms. In contrast, one of the proposed exercises in this work introduces this concept using a didactical approach to evangelize trainees about the inclusion of proactive defensive mechanisms to improve the security posture of the system against this type of attack.

3. Integration of Attack Detection Exercices in the SPIDER Architecture

5G networks rely heavily on virtualization technology, which allows agile service deployment and the promotion of the "slicing" concept, which is one of the focal concepts in 5G standardization forums (3GPP, ETSI, NGMN, etc.). However, in terms of security, the widespread use of virtualization radically increases the attack vectors exposed by 5G deployments. These attack vectors can be combined by hackers to manipulate part of the infrastructure. However, the specificities of the 5G environment are such (usage of NFVOs, SDN switches, and SDRs) that prevent the chance of a holistic training experience for a trainee.

SPIDER is an innovative 5G cyber range platform that bridges this gap by being able to address the functional requirements of security experts who want to verticalize in the specificities of a 5G environment. SPIDER delivers a next-generation and replicable cyber range platform for the telecommunications domain and its fifth generation (5G), offering cybersecurity emulation, training, and investment decision support. It features integrated tools for cyber testing, including advanced emulation tools, novel training methods based on active learning, as well as econometric models based on real-time emulation of modern cyber-attacks.

SPIDER's training activities are supported by four different learning modalities: Theoretical Training, Emulation Training, Simulation Training and Security Awareness Training through Gamification. The solution proposed in this paper was developed in the context of the Emulation Training modality, in which the trainees are asked to interact physically with a target deployed by SPIDER in a controlled environment, with the goal of hacking it (for red teams) or defending it (for blue teams). In the context of this modality, our solution specifically addresses the design and integration of defensive exercises to train blue teams on the use of ML-based toolboxes (Figure 1).

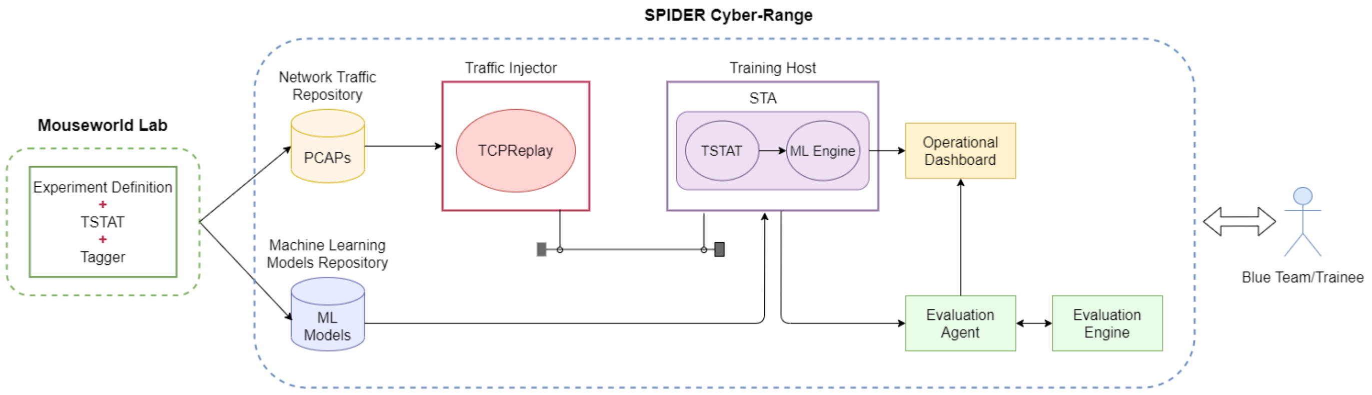

Figure 1. Integration of ML-based attack detectors in SPIDER defensive blue team exercises.

It should be noted that although the SPIDER cyber range has been designed as a generic platform that can address both offensive and defensive 5G security training and evaluation, in this work, we only focus on the defensive side of 5G security. Furthermore, SPIDER also supports penetration testing exercises to train the red team in the use of ML techniques to evade detection tools and discover new exploits in order to gain a deeper understanding of the security vulnerabilities that exist in the technologies and procedures involved in the deployment, testing, and operation of a virtualized 5G network. Because the focus of this work is on defensive exercises, we will henceforth use the terms "trainee" and "blue team" interchangeably in the text to refer to one or more individuals who are actively using the SPIDER platform to train and evaluate their defensive skills in detecting and mitigating attacks in a 5G network.

In the rest of this section, we detail the integration of SPIDER into defensive exercises designed to train a blue team on the use of ML toolboxes for the detection of network attacks. Although we selected two recently appeared attacks (cryptomining and DoH flooding) to exemplify the processes, it should be noted that other defensive scenarios can be integrated in a similar way.

Figure 1 details the architecture that we propose to integrate ML-based defensive exercises in SPIDER. From a general perspective, the proposed integration is based on the deployment in the SPIDER platform of a Smart Traffic Analyzer (STA), an ML-based toolbox with standardized interfaces that contains several ML models designed to detect an attack and a packet aggregator that groups packets into connections. During exercise execution, the STA helps the trainee discover a specific attack (e.g., a cryptomining or a DoH attack) more efficiently than traditional methods (e.g., rule-based systems). For example, if the trainee is trying to detect a cryptomining attack, they select and activate an ML model from those available in the STA with the aim of detecting the traffic connections responsible for the cryptomining activity. Similarly, in the case of DoH attacks, the trainee selects an ML model available in the STA to try to identify the traffic connections that are responsible for the DoH attack. The STA ML model provides the trainee with the predicted class for each connection and a score reflecting the model's confidence in its prediction. This allows the learner to monitor and identify traffic flows that are part of an attack in real-time. During the exercise, in the event that a new traffic flow is identified as malicious, the trainee receives a notification on an operational dashboard and is prompted to inspect the traffic and propose an action accordingly to effectively mitigate the attack.

The classification between normal and attack connections made by the ML models is the result of analyzing a set of informative features of each connection at a specific point in time. These features are generated by the packet aggregator that runs in parallel with the ML engine in the STA. The packet aggregator receives the packets that were generated previously during the emulation of the attack in real-time, groups them into connections, and computes the connection statistics at periodic time intervals. The aggregation of packets into flows (connections) generates both statistics related to the entire flow (e.g., flow duration, bytes exchanged, etc.) and statistics related to the packets exchanged in the flow (e.g., inter-packet delay). Different packet aggregators can be used for this task, such as the Tstat [31] and NetFlow [32] tools. The packets corresponding to an emulated attack have been previously stored in the Network Traffic Repository in a pcap file. These packets are sent to the STA component by the Traffic Injector, which reads them from the pcap file and reinjects them into an isolated part of the SPIDER network where the STA can read them. Note that the STA will receive the packets in real-time and in the same order as they were generated during the emulation of the attack. The output of the ML engine is sent to an operational dashboard (e.g., Kibana) to allow the trainee to observe the results and the performance of the ML model during the detection of the attack. Section 4 details the process conducted by the ML engine and the packet aggregator, and Section 7 explains the STA architecture that contains both components.

During the realization of the exercise, an Evaluation Agent is responsible for monitoring the progress of each trainee in the detection of the attack. At the end of the exercise, the Evaluation Agent communicates with the Evaluation Engine to assess the trainee's performance in the exercise. The final score that reflects the mastery that the blue team has achieved in the exercise is then made available on the SPIDER security operational dashboard as learning feedback. The feedback provided allows the blue team to identify specific areas for improvement and prepare for future exercises.

The attack and normal packets used in an exercise were previously generated by emulating the attack through the user plane of a virtualized environment of a 5G network in the Mouseworld Lab. In addition, the ML models running in STAs are also trained in the Mouseworld Lab. The Mouseworld Lab is an emulation environment of a network digital twin set up in Telefonica premises that allows the emulation of different attack scenarios in a controlled way and collects all packets generated during the attack emulation, both normal traffic and attack packets.

In the context of SPIDER exercises, the collected packets of an attack emulation conducted in the Mouseworld are stored in a pcap file to be used in two different contexts: (i) train ML models that can act as attack detectors in defensive exercises and (ii) reinject the packets into the SPIDER network during the emulation of an attack as part of a defensive exercise.

To prepare the Mouseworld Lab, the datasets required for training the ML models, the packets of an attack, are grouped into flow statistics using the same aggregator and in the same way as is in the STA during the execution of the exercise. Before flow statistics can be used in the training of supervised ML models, labels need to be added to each of them (e.g., a "0" to normal traffic flows and "1" to cryptomining flows). The annotation of flows is performed by the Tagger component in the Mouseworld Lab. Once annotated, the set of labeled flows can be used to train supervised learning models or as ground truth in unsupervised methods. Once trained, the ML models are included in the corresponding STA for the attack and exported to the SPIDER cyber range jointly with the pcap file that contain the packets of the emulated attack. The packets and the ML models corresponding to an attack are stored in the Network Traffic Repository and in the ML Models Repository, respectively, with both repositories acting as links between SPIDER and the Mouseworld Lab.

4. Machine Learning Processing Model in SPIDER Exercises

Figure 2 describes the ML processing model we adopted in the SPIDER cyber range. Using this processing model, the SPIDER platform can emulate a specific attack in an isolated part of the network within the SPIDER platform in two different ways:

- Deploying clients and servers that replicate a realistic scenario of the network attack and the normal traffic flowing through the network. In this scenario, packets are generated in real-time (A in Figure 2).
- Using network traffic previously captured and stored in pcap files that contains the attack and normal traffic packets. (B in Figure 2).

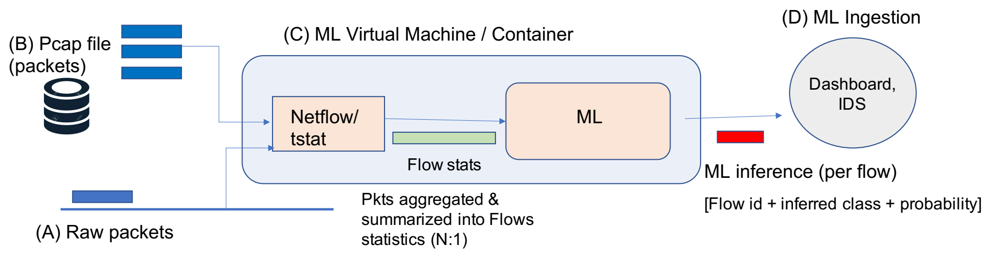
(B) Pcap file (packets)
(C) ML Virtual Machine / Container
(D) ML Ingestion
Dashboard, IDS
Netflow/ tstat
ML
Flow stats
ML inference (per flow)
[Flow id + inferred class + probability]
Pkts aggregated & summarized into Flows statistics (N:1)
(A) Raw packets

Figure 2. ML processing model in SPIDER.

The current version of the SPIDER cyber range only implements the second approach, and, therefore, the traffic containing the attack and normal traffic packets has to be generated externally in a separate system called the Mouseworld Lab.

The third component of the processing model is the ML Virtual Machine (C in Figure 2), which is composed of a packet aggregator that inputs connections statistics to a trained ML model. The last component of the processing model (D in Figure 2) deals with the processing of the ML output that can be consumed and processed in different ways, such as a monitoring dashboard or an Intrusion Detection System (IDS). For example, based on some preconfigured policies, the detection of a cryptomining connection that is above a certain confidence level causes an Access Control List (ACL) to be generated in order to block that connection at the ingress router of that connection. In the SPIDER platform, the output of the ML model will be sent to a dashboard so that the blue team performing the exercise can analyze the results in real-time and make a decision, which will be assessed during the evaluation of the exercise.

The ML models that are normally used in IDS and that we are going to integrate into the SPIDER platform exercises usually work at the flow level (i.e., TCP or UDP connections). Therefore, we will need to use a packet aggregator on connections to be able to extract the statistical features of these connections over time, both for training and validation of the ML models and to make inferences in real-time in the SPIDER platform exercises. These will be the statistical features of a connection that will be passed as input to the ML model so that the model can infer to which class the connection belongs. The output of the packet aggregator will be sent later as input to the ML model to perform the corresponding inference (e.g., the connection was detected as cryptomining or normal traffic with some probability). Assuming a non-excessively large volume of packets transmitted during the exercise, in SPIDER, we have adopted the open-source Tstat tool [31] to group packets into connections and extract the statistical features of the connections. Note that commercial packet aggregators (e.g., Netflow-based tools) can be used if the volume of packets to be processed is exceedingly large for Tstat.

The processing model described above is how attack detection scenarios are integrated into blue team exercises, in which different techniques not restricted to ML-based detectors are tested in response to a specific attack. For example, the exercise may lead the blue team to initially test traditional techniques (e.g., rule-based tools, such as Snort) and, if unable to detect the attack accurately, move on to test different ML models with varying degrees of training and hence accuracy and precision. The use of different ML models (or even the same model trained with different degrees of intensity) allows the blue team to learn that using a properly trained ML model is key if we want to detect attacks accurately and effectively, especially when dealing with encrypted malicious traffic.

The STA component was created to facilitate the integration of this ML-based processing model in the SPIDER cyber range. To that end, STA is based on container technology, contains a packet aggregator and an ML engine, exposes clearly defined interfaces, and allows the easy reconfiguring and parameterizing of the ML model running inside it. This component is detailed in Section 7.

5. Mouseworld Lab

The Mouseworld Lab [3], an emulation environment of a network digital twin set up in Telefonica premises, is used to set up and emulate attack scenarios in a controlled way and to generate and collect in a pcap file all packets of the attack and normal traffic to be used later in cyber range exercises and during the training and testing of ML algorithms. In SPIDER, when the exercise is started, these packets are reinjected into an isolated part of the SPIDER network to emulate the occurrence of the attack and normal network traffic (see Section 4).

The experiments that can be deployed in the Mouseworld Lab allow the capture, storage, and processing of network traffic representative of the attacks to be reproduced in the cyber range exercises. The processing that is performed on the network traffic captured in the Mouseworld Lab was initially oriented toward the training and validation of ML models for the detection of network attacks. To this end, the Mouseworld Lab provides a way to launch clients and servers and collect the traffic generated by them, even if they interact with clients and servers outside Mouseworld on the Internet.

This emulation environment allows configuring and executing specific attacks mixed with normal traffic instantiating virtual machines that deploy specific attack clients connected to real servers located at different points on the Internet. In addition, the emulation environment allows configuring other virtual machines on which normal traffic clients and servers (e.g., web, file hosting, streaming) are deployed. Finally, a commercial tool called BreakingPoint from Ixia allows a wide variety of realistic traffic types to be configured and injected into the network.

Once a configuration is deployed, all packets exchanged by the clients and servers with each other and with other servers on the Internet can be captured. The captures are stored in pcap format files in order to be used later for training and validation of ML-based attack detectors. Furthermore, these captures can be used in the SPIDER exercises following the processing described in Section 4 by reinjecting them into an isolated network segment that the SPIDER platform has reserved for the exercise. In this way, the blue team can solve the proposed exercises using (i) realistic network traffic and attacks stored in a pcap file and (ii) the corresponding ML models that were previously trained in the Mouseworld Lab with these network traffic data and that are leveraged as toolboxes for the SPIDER exercises. In this way, the blue team can experiment with different ML models (or the same model trained with different configurations of hyperparameters) and observe the system response to the attack in terms of performance and accuracy of the threat detection strategy. Finally, these captures can also be used to train the GANs component described in Section 8 to produce a synthetic data generator that can be used in cyber range exercises to generate different data sequences and thus avoid an exercise always running with the same dataset, as explained in this section.

The Mouseworld Lab has the ability to create or destroy different simultaneous scenarios and to launch different tests to generate traffic to be used for experimentation in a controlled environment. Based on the NFV/SDN architecture, it has the capability to create virtual scenario instances, isolate the traffic between scenarios in the experiments, generate network traffic on demand, and capture it (using VNF probes). One key feature of the Mouseworld Lab is the repeatability capacity, which allows us to evaluate different mitigation tools or versions in the same conditions and using similar statistical patterns. In the context of SPIDER, the combination of these characteristics allows the generation of realistic cybersecurity scenarios that can be used by the blue team to practice and gain confidence in the configuration and deployment of ML-based attack detectors.

6. 5G Traffic Generation

An essential part of the cyber range infrastructure is related to the possibility of emulating 5G networks and the correspondent traffic depending on the specific scenario being run in it. This means that the infrastructure must integrate the 5G network functions (NFs) that make up the core of this type of mobile network, and some mechanisms must be designed and deployed to generate and inject traffic through this network. It is possible to differentiate at least two types of traffic specific to 5G. On the one hand, the user-application traffic (i.e., traffic generated by standard applications in User Equipment, such as web browsers, streaming apps, etc.), and on the other hand, signaling traffic needed by the core Network Functions for the correct functioning of the network (that is, PDU session establishment, authentication processes, etc.).

To address the need to execute ML-based exercises in real 5G environments, a complete 5G network stack has been added to the Mouseworld Lab. This stack was designed using VNFs and, therefore, is compatible with the NFV management framework. Additionally, we developed custom software that is able to emulate a range of user equipment in terms of signaling traffic while allowing us to capture and inject any user traffic from generic sources in the 5G network. The rationale behind this signaling traffic generator/user traffic injector tool is to be able to generate a wide range of different traffic scenarios without needing the deployment of full User Equipment emulators or actual User Equipment with radio access and gNB base stations to receive the traffic.

As there will not be actual radio stations in the infrastructure, the software implements the protocols and functionalities between the stations and the network core, while the communications between User Equipment and gNB are not actually being emulated.

We describe in the following subsections the 5G traffic generation tools that we have designed and implemented in Mouseworld. In addition, we detail the 5G infrastructure deployment.

6.1. 5G Traffic Generation Tool

For the generation of traffic that can be used in scenarios that use the 5G infrastructure, we have designed and developed a tool that is able to inject signaling and real user traffic in the deployed 5G core, being able to use services located in the virtualized environment or in the Internet through the tool. The main goal of this tool is, consequently, to provide a way of injecting traffic into the 5G core network-virtualized infrastructure without dealing with actual Radio Access Networks and 5G-ready hardware. This tool acts as a signaling NAS traffic generator that is able to communicate with the AMF NF in the 5G core by emulating the operations of User Equipment in a real environment. The tool can perform session management, UE registering and registering, etc. Additionally, it can maintain tunnels that use the GPRS Tunneling Protocol (GTP) with a UPF in the core network and send data through them. The user data to be sent through this tool, which acts as a broker, is captured in a network interface and can have any virtual appliance or hardware machine as a source as long as they are connected to the tool interface.

As there are no actual radio stations in the infrastructure, the software implements the protocols and functionalities between the stations and the network core, while the communications between User Equipment and gNB are not actually being emulated.

Figure 3 shows a simplified diagram of the traffic generation architecture. It is based on the deployment of a virtualized 5G core in the cyber range infrastructure, composed of the main Network Functions. The traffic generation software is then connected using virtual network interfaces with the Network Functions that are usually connected to the gNB base stations through the N2 and N3 interfaces (connected to the Access and Management Mobility Function (AMF) and the User Plane Function (UPF)).

The traffic generation software is deployed in a virtual instance within the virtual infrastructure of the cyber range, and it is equipped with as many virtual network interfaces as needed to connect other virtual instances. These virtual instances would be in charge of running network applications (web browsing, video streaming clients, traffic generation clients, etc.) and, therefore, would act as the users of the 5G network in the emulated environment. The traffic generated by these instances is captured and processed to be sent to the 5G network. This means that, for each emulated User Equipment, a new session is established, a new tunnel ending point is added, and all traffic is encapsulated using a GTP-User Plane (GTP-U) tunnel. The encapsulated traffic is sent to the UPF Network function from the traffic generation software, and then it is routed to its destination.

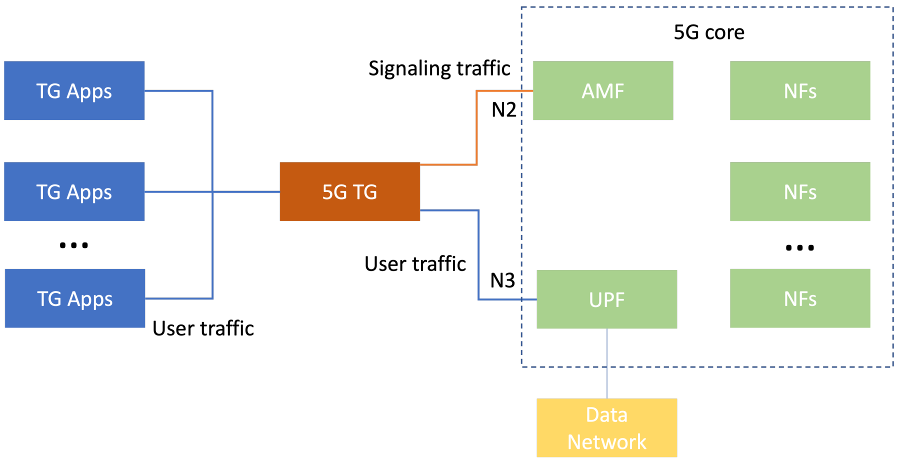

Figure 3. 5G traffic generation architecture.

6.2. 5G Infrastructure Deployment

Both the NF implementation and the traffic injection tool have been deployed and integrated into the Mouseworld environment. The goal of this integration is to provide the project with the required 5G infrastructure for the execution of dataset generation tests related to specific ML scenarios. These use cases aim to provide mechanisms in exercises for the detection of attacks on the 5G infrastructure by hijacking instances in the 5G core or by injecting malicious traffic through it.

The 5G core network NFs have been deployed using Docker containers inside a single OpenStack instance using Docker bridge networking for the internal communications between each module. Then, the Docker interfaces have been mirrored, allowing for the capture of all the traffic (signaling and user traffic) that is shared outside and inside the core network.

Figure 4 shows the Mouseworld integration for the 5G infrastructure. Mouseworld is composed of various virtual appliances that act as clients or servers for different types of traffic (video streaming, web pages, etc.). It also counts on an orchestrator that communicates with all appliances through a management network and the possibility of monitoring and capturing traffic for analysis, statistics, and dataset generation. It is also connected to remote services through the Internet.

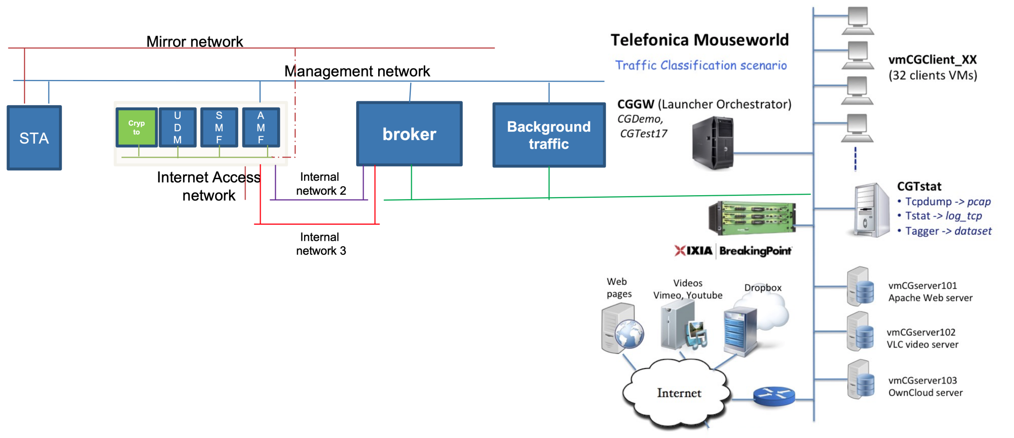

Figure 4. 5G core infrastructure and traffic injection tool (broker) integration in the Mouseworld Lab.

The new appliances added to the existing infrastructure are in red in the figure. The 5G core infrastructure has been integrated into a single instance, using Docker containers, as explained before. Virtual links have been added to communicate the required containers (specifically the AMF and the UPF) with other appliances in the environment. The UPF is connected to an internal network that allows for the user traffic to reach the local or remote servers through the Internet.

On the other hand, the traffic injection tool has been deployed in a different appliance (shown as the broker in Figure 4) that is able to communicate with the AMF using signaling traffic and is able to receive users' traffic from the clients in the Mouseworld environment. The traffic is tunneled and sent through the link connected to the UPF container in the Dockerized 5G core.

The deployed 5G infrastructure has been adapted to the requirements of the scenarios that are going to be tested using it, namely the cryptomining detection and DNS over HTTPS (DoH) infrastructure attacks scenarios. However, the 5G infrastructure and connectivity service are generic enough to suit different 5G network security exercises.

Additionally, any potential scenario may require the generation of different types of traffic from the 5G clients, including background traffic, to be captured along the use case-related traffic. Additional clients, linked with 5G session registering in the 5G core control plane, are supported, using traffic generation tools for the generation of background traffic to be injected into the 5G core network.

7. Generation of ML Toolboxes for SPIDER Exercises: Smart Traffic Analyzers

The ML infrastructure described in Figure 2 is deployed in a common framework, named the Smart Traffic Analyzer (STA) toolbox, to deliver different ML tools into different scenarios and exercises. This component acts as the link between the Mouseworld Lab and SPIDER exercise scenarios. Once a new scenario is deployed and trained in the Mouseworld Lab, then a new STA is produced and transferred to the SPIDER infrastructure to be used in the corresponding exercises.

The solution depicted in Figure 5 shows how several tools (add-ons) are supported for network traffic analysis exercises related to ML. The main component is the STA Service, the internal logic and pipelines were introduced in Section 4, and the goal is to provide a unique artifact as a CNF (Docker) that allows the use of different Machine Learning models as well as the modification of the configuration files in order to adjust them to each scenario. The current version of the STA service is customized by two files containing (i) the list of features to be input to the ML model and (ii) the list of tags identifying the classes of the ML output (e.g., crypto or normal traffic). ML model swapping can be performed when starting or restarting the STA, indicating the location where the new ML model is. Furthermore, STA integrates the capacity to configure the endpoint in the SPIDER dashboard to deliver the classification events obtained while performing the exercises.

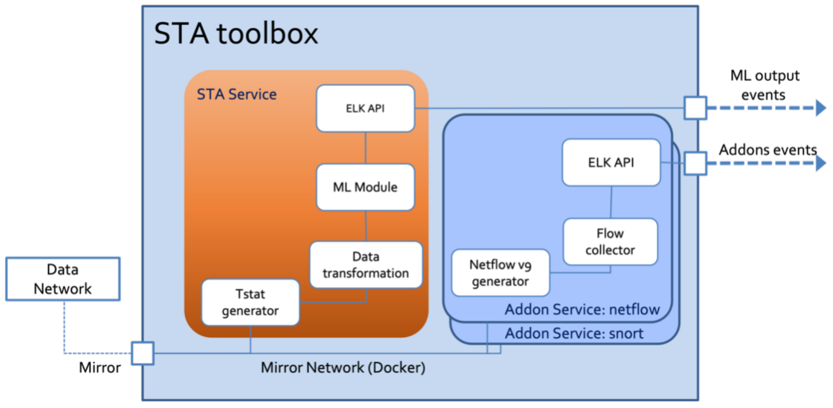

Figure 5. STA toolbox with two add-ons.

The STAs are based on machine and deep learning models and can analyze network flows (classify, predict, detect anomalies, etc.) without accessing the packet payload. STAs can be applied in real-time scenarios (i.e., processing the packets received from the data network) or as forensic tools (reading the packets contained in a pcap file).

The generation of an STA consists of four modules interacting in a pipeline in the Mouseworld Lab:

1. A topology generator built on top of OSM (Open Source MANO) and OpenStack;
2. A Launcher component for deploying and running experiments;
3. A Tagger component that collects the network traffic generated by the Launcher and labels it in an automated way and without expert intervention;
4. A Machine Learning Model Builder that encompasses the training and validation processes of machine and deep learning models using the labeled datasets generated by the Tagger as input.

In addition, a commercial tool from Ixia called BreakingPoint is integrated into the process to enrich the traffic generated with a diversity of Internet protocols.

Figure 6 describes the traffic generation and the gathering and labeling processes:

1. The Launcher uses a customer network specification as input and runs experiments in client machines that generate real network traffic that crosses not only the Mouseworld network but also the Internet. Additionally, and with the aim of mimicking the statistical distribution of Internet traffic patterns, the Launcher runs synthetic sessions that generate network traffic from a collection of complementary Internet protocols using Ixia Breakingpoint, a commercial tool that allows the generation of complex patterns of synthetic traffic. The injection of these packets into the network is made in parallel with the network traffic generated by the real clients and servers.
2. The packets transmitted in the network are collected using the tcpdump linux tool and stored in pcap files. Then, the Tstat tool is used to group packets into flows (based on the five-tuple of source and destination ip-address/port number and transport protocol) and extract the corresponding statistics from them (e.g., number of packets sent from client to server). Next, the Tagger adds, automatically and without human intervention, labels to each flow using the log information output by the Launcher during the execution of each experiment (e.g., the destination IP and port of a cryptomining server are used to add a "crypto" label to all connections that match these two values). In the last step, using as input the labeled datasets output by the Tagger, the training and testing of supervised machine learning models are conducted.

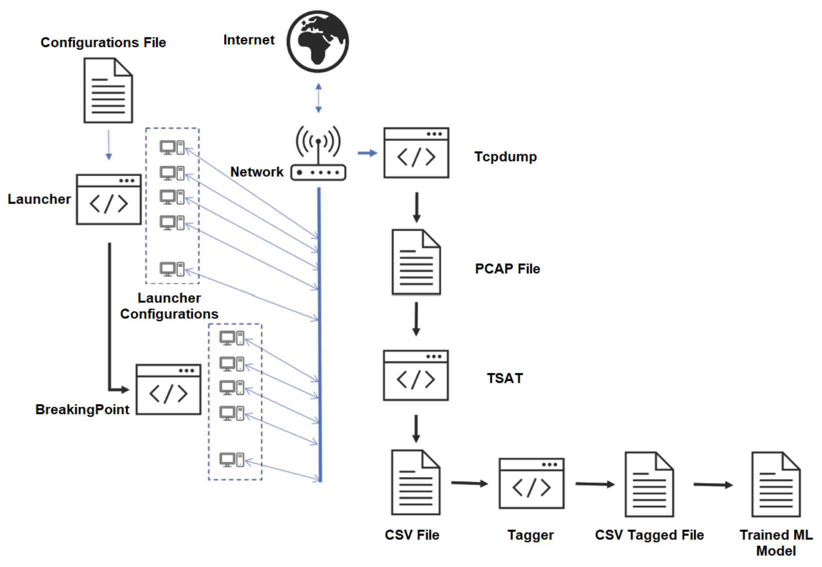

Figure 6. Traffic generation and labeling.

To complement this explanation, we detail the realization of a cryptomining attack scenario on top of the Mouseworld Lab. This scenario, based on the generic setup of Figure 4, will allow us to generate network traffic corresponding to normal connections and cryptomining ones. After collecting all packets in a pcap file, the STA module will be generated following the process described above. Both the pcap file and the STA module will be stored in the SPIDER cyber range to be used later in the cryptomining exercises. With the exception of the Machine Learning Model Builder, which is not represented in the figure and can be run on a separate infrastructure, the rest of the components for the generation of the cryptomining STA are deployed and run in the Mouseworld Lab. Note that the training and testing processes of the Machine Learning Model Builder, which tend to involve the execution of complex deep neural network architectures, are performed more efficiently in GPU-based infrastructures. Therefore, this last stage of the process can be executed on a more specialized computation infrastructure. When the training and testing processes are finalized, the resultant model is exported as a file using a model-specific format. The file containing the model will be stored later in an STA. Recall that STAs can be configured with several ML models that can be selected and activated on demand, one at a time, during the execution of the STA. Although several file formats are currently available to store ML and DL models (e.g., ".h5" files using the Hierarchical Data Format (HDF), and ".pkl" files using the Pickle serialization model), we strongly suggest using secure and open formats, such as ONNX [33].

8. Generation of Synthetic Traffic with Generative Adversarial Networks

In this section, we detail an innovative solution that applies a recently appeared generative model, named Generative Adversarial Network (GAN), to generate synthetic on-demand flow-based network traffic that (i) mimics network attacks and normal traffic and (ii) can be seamlessly integrated and used in SPIDER exercises.

A Generative Adversarial Network (GAN) [34] is a generative model in which two neural networks, G, called the generator, and D, called the discriminator, compete to improve their performance (Figure 7). From random noise provided as input, the generator learns to generate new data that are statically similar to the real data, while the discriminator learns to separate the generated data from the real data. The competition appears because networks D and G try to improve non-simultaneously satisfactory objectives. On the one hand, D tries to improve its performance in the classification problem, but on the other hand, G tries to generate as best results possible to cheat D. This adversarial relationship is the fundamental feature that allows the GAN generator to produce realistic synthetic data.

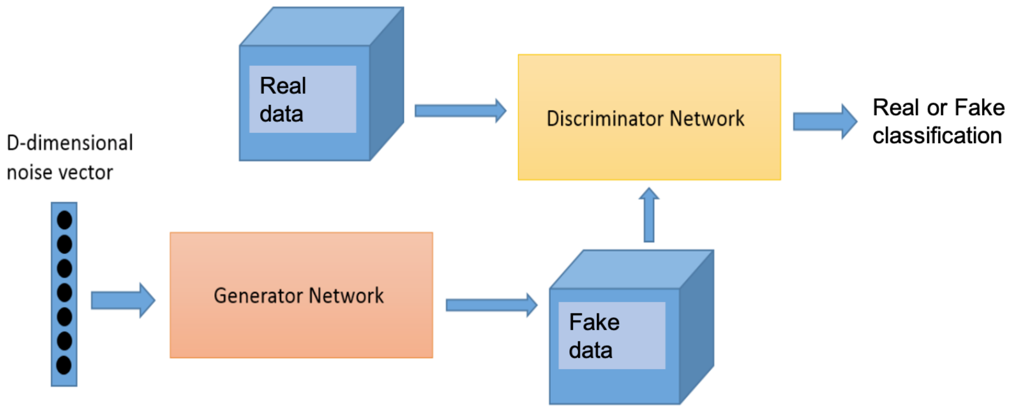

Figure 7. GAN architecture.

After training a GAN, and due to the dimension of the noise vector, typically 100, the generator can produce a theoretically infinite number of samples that follow the same statistical distribution of the real data. Therefore, using a GAN per type of traffic (e.g., one GAN for normal traffic and another for attack traffic), we can obtain as many samples of each type of network traffic as needed.

Our GAN solution is based on a state-of-the-art GAN named Wasserstein GAN [35], in which we substitute the activation functions of the output layer of the generator with new activation functions based on the Smirnov probabilistic transformation [36]. This innovative modification of a vanilla WGAN allows us to significantly improve the quality of the generated data regardless of whether the data distribution is continuous or discrete. In contrast to other existing approaches in the literature that are based on data augmentation solutions, our GAN solution generates synthetic data with such high fidelity that synthetic data can fully replace real data (attacks and normal traffic) in ML training processes, obtaining equivalent performance when the resultant models are tested with real data. This solution has two clear advantages: First, we address the existing shortage of publicly available network traffic datasets containing attacks and normal traffic, and second, we avoid the privacy violations that could appear when real data are used in ML/DL training and testing processes. In [36–38], the reader can find all the technical details of the GAN techniques we applied to our solution.

Besides the use of our GAN solution for training and validation of ML models, synthetic network traffic and attacks generated by our GANs can be used in cyber range exercises to generate different data for a particular type of exercise and prevent blue teams from always learning the exercise with the same data. In this way, a learner can repeat the same exercise using different data instead of always working with the same set of labeled traffic, which brings no advantage from a learning point of view. For example, having trained a GAN model to replicate a given type of attack (or normal traffic), we can generate as many attacks of such type as required. Therefore, even if the blue team repeats an exercise several times (rounds), the analyzed attacks and normal traffic are not going to be exactly the same in each run of the exercise.

In addition, red teams can also use our GAN solution in penetration test exercises to generate realistic attacks that never contain the same attack data, even if the launched attacks are of the same type. Thus, the robustness of an IDS against a type of attack can be evaluated by launching many different synthetic samples of the same attack.

Finally, a cyber range can import datasets from third parties containing attacks and normal traffic that are subject to privacy or anonymity restrictions. As the network data used in the exercises by the blue and red teams are the synthetic ones generated by our GAN solution, no breach in privacy appears during the realization of such exercises. Moreover, exporting attacks and normal data (e.g., to other platforms in a federated cyber range) can be performed without incurring any privacy violation as the exported data to be shared with a third entity are exclusively the synthetic network traffic generated by our GAN solution.

In Figure 8, we detail the way in which the synthetic traffic generated by our GAN solution can be integrated into SPIDER exercises and, in particular, in the ML processing model. Currently, we have developed and trained GANs for defensive exercises that can replicate with high fidelity the flow statistics of network attacks and normal connections. As can be seen in the figure, the GAN generator directly feeds the ML engine (solid red line). It is worth noting that from a functional perspective, the trainee (blue team) will not observe any difference during the exercise between receiving synthetic or real flow statistics since the ML engine will receive the input data with the same format either way. Nevertheless, future versions of our GAN solution (dotted red line) will be able to generate realistic sequences of network packets (attacks or normal traffic) that will be fed to the ML engine (contained in the STA module) in the same way as we feed the ML engine using a pcap file.

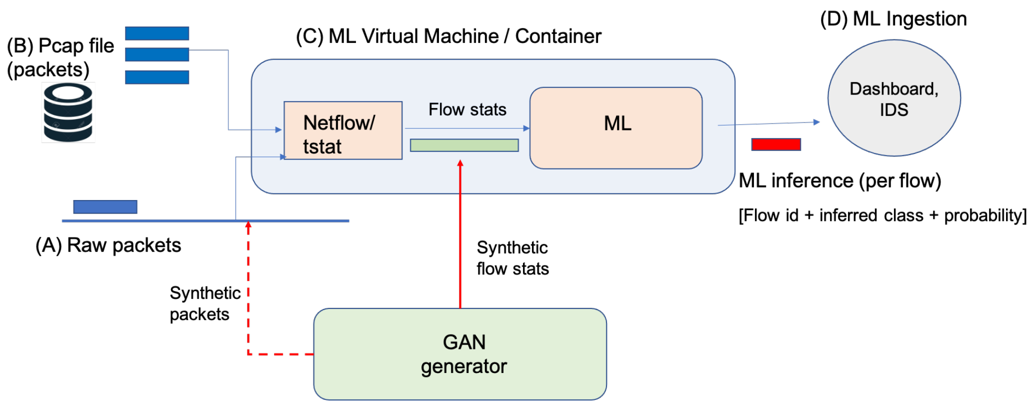

Figure 8. Integration of GAN synthetic traffic into ML model.

9. Exercises

SPIDER training exercises consist of instructing the trainee in the use of ML tools to detect different types of cyber-attacks, with a focus on the correct interpretation of model results to enable the development of successful strategies to effectively mitigate ongoing attacks. During the training process, the trainee will be able to evaluate the detection performance of a variety of ML models and classical rule-based methods and compare the results between them in the SPIDER dashboard. Specifically, in an attack scenario, the trainee acts as a member of the blue team and is tasked with obtaining the confidence values of several ML models in detecting attack traffic in order to choose the most effective one for the task. In addition, during the process, the learner is asked to answer some questions to assess the learning outcomes.

In order to teach specific detection methods for two different attacks (cryptomining and DoH flooding attack detection), two separate exercises have been created. Both exercises include a guide to follow, which contains instructions, tips to help solve the proposed assignments, questions to reflect on the topics, and tasks to complete. At the end of each session, the user will be able to observe the final result along with all the tasks that they completed correctly.

It is important to note that both scenarios rely on two command line interfaces that allow remote execution of commands on the designated virtual machines that will serve as the training environment. One of the terminals will be used to inject traffic into the other, which will process it in different ways depending on the task at hand. The two exercise scenarios are described in the following subsections. In addition, in the last subsection, we explain the details of the performance evaluation of the exercises executed by the blue team.

9.1. 5G Cryptomining Attack Detection Scenario

In this section, we describe the cryptomining attack detection scenario that occurs in the control plane of a 5G network. First, we introduce the cryptomining attack. Next, we present details on the integration of the ML/DL models into the training scenario for this particular attack. Then, we describe the proposed exercise to train cybersecurity professionals in the use of traditional and more advanced ML/DL-based methods to detect cryptocurrency attacks on a 5G network in real-time.

9.1.1. Cryptomining Attack Description

One of the most discussed threats to 5G is the possibility of leveraging the computational resources provided by the network to perform cryptomining attacks. Cryptomining is a process of validating transactions on a decentralized cryptocurrency blockchain. The attack works by creating a botnet of devices, called miners, which are used to validate transactions and receive rewards in the form of digital currencies, such as Ethereum (ETH) and Monero (XMR).

The attacker can use devices that are already infected with malware or can infect new devices, hijacking their resources to create the botnet for mining. Attackers can use various ways to infect devices, such as spreading malicious links on social networks, using phishing attacks, or spreading malicious applications.

In addition, the attacker has to choose the cryptocurrency to be mined and the mining pool that they want to join to validate transactions. A mining pool is a service that allows miners to combine their resources to validate blocks of transactions and receive rewards in exchange. Once the attacker has collected all the pieces, they can set up the botnet to mine cryptocurrencies for the criminal's benefit.

To detect cryptomining traffic in a timely and accurate manner, the network is the best place to identify these types of attacks [39]. However, detecting cryptocurrency mining activity on the network can be challenging, as encryption methods have been adopted in most of today's network protocols. For example, the attacker can use the SSL/TLS encryption protocol to hide the cryptomining protocol in the payload of the encrypted communication. For this reason, classical techniques of Deep Packet Inspection (DPI) or identification of mining pool domain names (in the case of using encrypted SNI or web proxies) are ineffective in detecting mining activity in today's networks and more sophisticated techniques are needed to prepare today's cybersecurity professionals to deal with these problems in real-life situations.

To effectively mitigate these threats, ML/DL techniques can be used to train models that can accurately identify the presence of cryptomining traffic in real-time, even if it is encrypted, using a set of salient flow features collected at the network and transport levels [39]. In this exercise, we ask the student to attempt rule-based detection of the presence of encrypted and non-encrypted cryptomining traffic on the network using traditional NIDS, such as Snort. Afterward, we ask the student to take advantage of an ML/DL-based cryptomining detector provided as part of the STA module to more effectively identify the presence of cryptomining traffic on the network.

9.1.2. Cryptomining Attack Detection Using Machine Learning

Detection of cryptocurrency mining activity has traditionally been performed by identifying system or network anomalies that are not usually present in a normal situation and may be indicative of some form of attack or compromise. In the case of system anomaly detection, it is common to use features such as CPU, GPU, and memory usage and power consumption metrics to detect unusual activity on systems that may be related to the specific kind of attack under study [40]. However, this approach is not always effective, as cryptocurrency mining activity may be conducted in such a way that avoids raising the alarm to system administrators, and it is also not always applicable, as monitoring access to internal systems is not possible in many cases due to privacy or security concerns. In the case of detecting cryptocurrency mining activities at the network level, anomaly detection focuses on network traffic to identify suspicious activity, the most common method being DPI (Deep Packet Inspection). DPI is a well-known technique for detecting network anomalies by intercepting, inspecting, and analyzing network traffic with protocol-specific detection engines that look for known patterns in the payload, abnormal bandwidth consumption, network communication anomalies, or the use of unusual ports or types of traffic. However, DPI is not very effective in detecting cryptomining activity since traffic is usually encrypted, and even in the case of unencrypted traffic, relevant information such as traffic type and packet content can be disguised using various cryptographic techniques implemented at the application layer.

Consequently, recent approaches to detecting cryptocurrency mining activity have been geared toward using ML/DL algorithms to characterize and detect unusual patterns in network behavior that may be indicative of cryptocurrency mining. However, to date, few studies have been conducted to detect cryptocurrency mining activity at the network level using this approach. Among them, a study conducted by Muñoz et al. [41] evaluated four different ML techniques: Naive Bayes, Support Vector Machine, CART, and C4.5

to detect cryptomining activity for five different cryptocurrencies (Bitcoin, Bitcoin-Cash, DogeCoin, LiteCoin, and Monero) from unencrypted cryptocurrency flows. This work presents an alternative to the use of DPI that uses eight different features derived from NetFlow/IPFIX metrics to detect cryptocurrency mining activity from network traffic. Therefore, the proposed approach does not require payload access, and only network flow statistics are used to perform the task. In their results, they showed that the CART model was able to distinguish cryptocurrency flows from normal traffic with high precision while being able to identify the cryptocurrency being mined. It should be noted that the identification of the particular cryptocurrency being mined by the attacker would only be effective in the real environment for the five different cryptocurrencies that were considered in this study, and the results for other types of cryptocurrency mining activities would not be correct and cannot be relied upon. Considering the diversity of the cryptocurrency mining landscape in terms of the number of available cryptocurrencies, this would inevitably cause uncertainty in the identification of the cryptocurrency being mined and, therefore, the utility of this method is limited for actual use in the real environment. However, despite the positive results obtained in this study in terms of prediction accuracy, it is important to remark that the work presented by Muñoz et al. is limited only to unencrypted Stratum protocol traffic, which represents only a small fraction of mining activity, and, therefore, their approach lacks utility for real-life applications.

In contrast to the work presented by Muñoz et al., the approach followed by Pastor et al. [39] uses a much larger feature set, consisting of a total of 51 informative features that were extracted using the Tstat tool. In addition, more complex models, such as Random Forest and Deep Neural Networks, were applied to increase the accuracy and improve the identification of cryptocurrency mining activity. More importantly, encrypted traffic was also taken into account in this work, providing a more realistic approach to detecting cryptocurrency mining activity at the network level for real-life scenarios.

Following a more traditional direction, Swedan et al. [42] proposed a method to detect and block unencrypted cryptocurrency mining connections originating from a web browser when the user visits web pages that have embedded third-party resources dedicated to performing cryptocurrency mining as a background activity while the user navigates (e.g., CoinHive, Crypto-Loot). Their approach is based on the deployment of a proxy server within the local network and the use of DPI to analyze real-time traffic and detect and block web pages calling cryptomining scripts. However, this approach has several disadvantages. First, it does not cover all types of mining traffic, as it only applies to traffic generated by the web browser and, therefore, considerable cryptomining activity may go unnoticed. Second, using DPI to analyze real-time traffic is a very demanding process that requires a large number of computational resources to be effective and also fast enough to enable the real-time detection of cryptomining activity. Third, the use of DPI to analyze real-time traffic can negatively affect the user experience when browsing the web if a high number of false positives are triggered by the detection mechanism. Finally, this approach is not applicable to encrypted traffic, which, as mentioned above, represents a considerable percentage of traffic on today's Internet.

The method we apply in this exercise follows the same approach proposed by [39]. We train complex ML models using a large set of network features obtained from flow statistics to perform cryptomining activity detection for both non-encrypted and encrypted connections with a high level of accuracy.

9.1.3. Application and Integration of Machine Learning for Cryptomining Attack Detection

To train our ML models, we used the tagged datasets that were described in Section 7. In this case, each row of the dataset was tagged as either 0 (normal traffic) or 1 (cryptomining attack traffic) using the IPs and ports of the known attack connections. Once we had access to this dataset, we used Python's sklearn library [43] to train a Random Forest Classifier to predict whether a connection corresponds to cryptomining activity or not according to all the features derived from the Tstat statistics, except IPs and ports, as they are used to label the dataset (class labels) and, therefore, cannot be used to train the model. Although the accuracy of the model using all these features is already high, many of these features do not contribute significantly to the prediction and can be ignored to improve the training and inference efficiency of the model. Therefore, we decided to make a random selection of commonly used features and managed to reduce the input needed to ten features while only reducing the F1-Score by less than 5% in our testing data. In this exercise, we selected only ten features in contrast with the DoH exercise, where we chose a large number in order to show the trainee that ML models can use different sets of features even when they are going to be applied to the same task. The ten features used for the generation of this model are listed in Table 1. Note that some features are sometimes separated into Client to Server (CS) and Server to Client (SC) communication. These features appear in the table with different CS and SC identifiers.

Table 1. Tstat selected features for cryptomining attack detection.

CS	SC	Name	Metric	Description
3	17	packets	-	Total number of packets observed from the client/server
5	19	ACK sent	-	Number of segments with the ACK field set to 1
7	21	unique bytes	bytes	Number of bytes sent in the payload
8	22	data pkts	-	Number of segments with payload
9	23	data bytes	bytes	Number of bytes transmitted in the payload, including re-transmissions

For didactic purposes, we decided to create three different models with varying levels of precision, which could teach the blue team that ML tools are not fool-proof and can sometimes offer wrong results if trained incorrectly. The hyperparameters that we decided to modify in this case were the number of estimators and the maximum depth of each tree. The Random Forest Classifier creates a number of decision trees with a certain depth that is used to infer a class. The number of decision trees that the algorithm will create is controlled by the n_estimators attribute, while the maximum depth of each decision tree is controlled by the max_depth attribute. As these numbers become smaller, the algorithm will have fewer decision trees to contrast an estimation and will be able to create decision trees with less complex solutions. Therefore, both of these parameters can negatively impact the model performance if they are not correctly adjusted for the problem at hand. A manual selection of values for these hyperparameters was performed to obtain a variety of models with different performance levels. We list these values in Table 2.

Table 2. Hyperparameters for each cryptomining Random Forest Classifier model generated

Model Name	Max. Depth	Num. Estimators
crypto_spider_5g_rf_10_a.pkl (Good performance)	Unlimited	100
crypto_spider_5g_rf_10_b.pkl (Medium performance)	8	3
crypto_spider_5g_rf_10_c.pkl (Bad performance)	4	1

The quality measures of Table 3 show how the variation in the hyperparameters affects the model's precision and the generation of false positives and false negatives. The quality measures we have taken into account are the F1-Score, the balanced accuracy (which is an accuracy metric that takes into account the unbalanced classes of our dataset), and the confusion matrix.

Table 3. Quality measures of cryptomining traffic classification models. In the confusion matrices 0: NORMAL, 1: CRYPTO, where rows represent true values and columns predicted values.

Model Name	**Quality Measure**	**Score**		
crypto_spider_5g_rf_10_a.pkl (Good performance)	F1-Score	0.954		
	Balanced Accuracy	0.956		
	Confusion Matrix		**0**	**1**
		0	421,968	0
		1	144	1494
crypto_spider_5g_rf_10_b.pkl (Medium performance)	F1-Score	0.822		
	Balanced Accuracy	0.855		
	Confusion Matrix		**0**	**1**
		0	421,940	28
		1	475	1163
crypto_spider_5g_rf_10_c.pkl (Bad performance)	F1-Score	0.463		
	Balanced Accuracy	0.678		
	Confusion Matrix		**0**	**1**
		0	421,669	299
		1	1054	584

The results of Table 3 also reflect the didactic value of this exercise. By testing real traffic against these different models, the blue team will be able to easily observe the contrasting results shown by a decrease in result confidence and inconsistency in detecting attacks in the worse models. This exercise thus provides a scenario that is very similar to a real-life situation, proposing the evaluation of different ML models to check their performance in practice and allowing the end user to judge which one is best suited to their needs. In this way, we teach the student how to extract the full potential of this technology while highlighting the possible pitfalls that can occur when it is applied incorrectly.

9.1.4. Description of the Cryptomining Attack Exercise

The scenario consists of two VMs communicating with each other, as shown in Figure 9. The first VM uses the tcpreplay tool to replay real network traffic from a given PCAP file. This network traffic belongs to existing signaling communication in the 5GCore based on the use of service-based interfaces (HTTPS) and is generated by user activity in the 5G network (login, registration, deregistration, handover, etc.). Furthermore, there is traffic related to cryptomining activity by an infected component on 5GCore. Network traffic is generated and stored in a PCAP file in the Machine Learning Lab for this exercise. The second VM consists of the three components described below.

- softflowd component: This component receives network traffic, processes it, and sends it to the ELK component for display.
- Snort component: This component uses a set of rules to identify unencrypted cryptomining connections from the received network traffic. It also sends alerts to the ELK component.
- STA Machine Learning component: This component uses the Tstat tool for feature extraction, processes the obtained data, and makes inferences for each connection. It also sends the labeled connections to the SPIDER control panel for visualization.

The trainee will have to use each of these components effectively to solve the proposed exercises. All the phases and processes the blue team needs to perform to complete the cryptomining exercise are described in Figure 10. In the following paragraphs, we describe the exercise and expand upon its different phases.

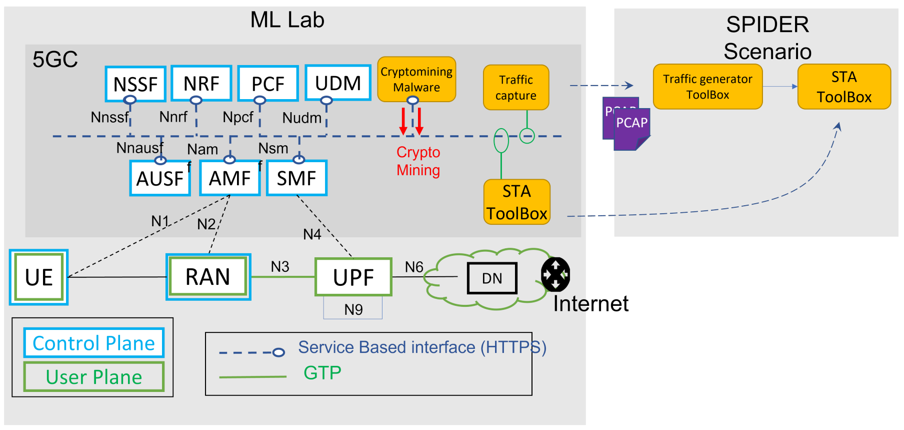

Figure 9. Cryptomining discovery scenario.

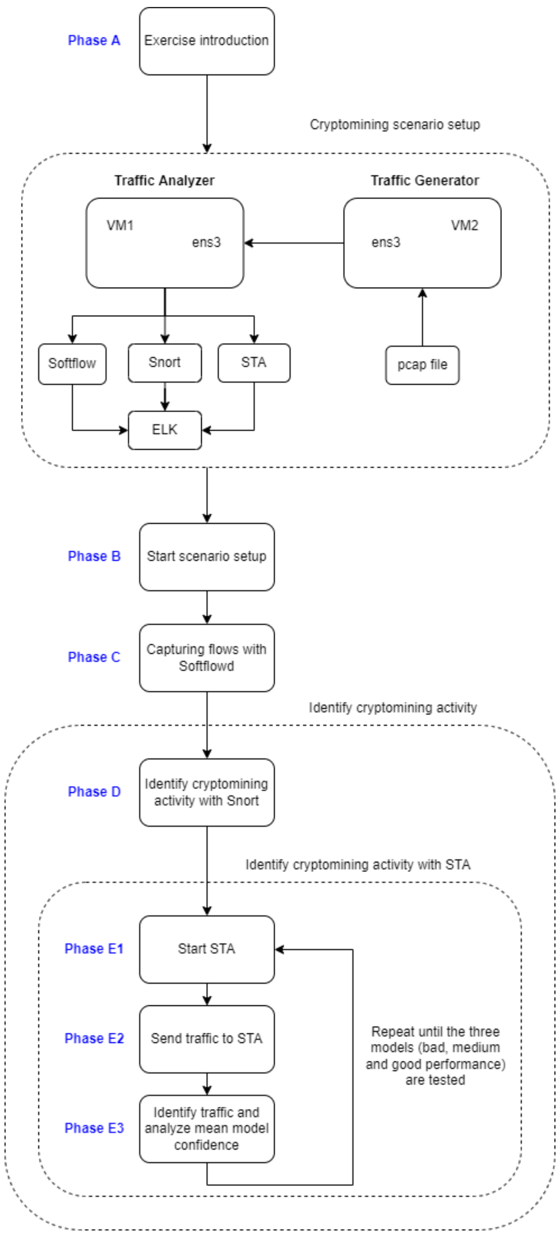
Phase A
Exercise introduction
Cryptomining scenario setup
Traffic Analyzer
Traffic Generator
VM1
ens3
VM2
ens3
Softflow
Snort
STA
pcap file
ELK
Phase B
Start scenario setup
Phase C
Capturing flows with Softflowd
Identify cryptomining activity
Phase D
Identify cryptomining activity with Snort
Identify cryptomining activity with STA
Phase E1
Start STA
Phase E2
Send traffic to STA
Phase E3
Identify traffic and analyze mean model confidence
Repeat until the three models (bad, medium and good performance) are tested

Figure 10. Cryptomining scenario phases.

In the exercise introduction of Phase A, the blue team is asked to assume the role of a security expert in the SOC (Security Operative Center) of a 5G cloud infrastructure provider. In this scenario, the user receives a ticket from the 5G operator team, reporting that they are experiencing performance problems in their service. The user must, therefore, analyze the traffic provided to them, detect the compromised devices, and identify the malware in the form of encrypted and non-encrypted cryptomining attacks that are impacting the performance. To that end, the user is asked to follow steps that include different detection strategies, including Netflow analysis and the STA component with several inference engines customized for cryptomining detection. Before starting the exercise, as represented by Phase B, the blue team is asked to execute a command that will reset all logs and dashboards of the scenario. In that way, the exercise can be performed in a clean environment. The blue team will also have access to this command to reinitialize the environment and restart the exercise from the beginning in case it is needed.

Part 1: Capturing Flows with Netflow

The first part of the scenario contains the instructions for analyzing 5G cryptomining traffic using the softflowd tool, as depicted in Phase C. The blue team will be tasked with starting a container with the tool using a command prepared on the first machine, which we will refer to as VM1. Once the container is active, the blue team will be able to inject 5G traffic into it, which contains a mixture of normal and cryptomining attack traffic. To do this, the blue team will run another ready-made shell script on the traffic-generating machine, which from now on, we will call VM2. After executing the traffic injector command, the blue team will be able to visualize the generated statistics on a Kibana dashboard. Using the statistics shown in the Kibana dashboard, the blue team is tasked with extracting some basic information regarding the traffic distribution, such as the most used ports. The blue team will also be able to use different filters to explore the characteristics of the traffic using different features, such as IPs or protocols in an attempt to distinguish between normal and cryptomining traffic. However, the main goal of this section is to lead the blue team to the conclusion that this information is not enough to monitor and protect a 5G network, as the softflowd tool merely provides basic information about the current traffic, such as origin and destination IPs and ports, which cannot be directly used to infer a cryptomining attack automatically. The next section will help teach the blue team how to identify the cryptomining activity.

Part 2.1: Identifying Cryptomining Activity with Snort

In this section, which corresponds to Phase D, the blue team will be tasked with identifying cryptomining attacks using the Snort tool in VM1. This tool is provided in a different container, and a different command is made available to start it. The blue team will have to send traffic to the tool from VM2, which can be performed in the same way as described in the previous section. Back in the Kibana dashboard, the blue team is tasked with observing the alerts generated by Snort. However, after performing this step, the trainees will discover that Snort does not display any alerts because the rules are intentionally not correctly specified. The reason is to familiarize the blue team with the format of Snort rules and to get them to invest effort in researching their correct specification. The blue team is responsible for modifying existing rules to allow Snort to correctly identify cryptocurrency traffic and send alerts to Elasticsearch. After researching the resources provided in the instructions, the blue team will be able to edit a specific configuration file and lay out the rules to detect cryptomining connections from Monero pools. The blue team will then be able to test their rules and observe the alerts displayed in the Kibana dashboard. Using the results, the blue team is then asked to provide the IPs that have been identified by Snort as belonging to Monero pools and thus being flagged as cryptomining attacks according to the specified rules. The blue team is then asked to reflect on the type of traffic with which Snort can be used and will come to the conclusion that this tool will be completely ineffective in detecting attacks that use encrypted traffic. As a solution to this problem, the next step discusses the use of the STA component for the identification of both encrypted and unencrypted cryptomining traffic.

Part 2.2: Identifying Cryptomining Activity with STA

In this step of the exercise, the blue team will be provided with three different ML models, trained with the same dataset to detect cryptomining activity. The difference between these three models is the hyperparameters used to train them, which will greatly influence their performance and the confidence of their predictions. In Phase E1, the blue team will be tasked with starting the STA containers in the VM1 with the provided command and restarting it with a different model each time. In Phase E2, by sending traffic from the VM2 using the methods described in the previous sections, the STA container will be able to analyze it. In the Kibana display, as part of Phase E3, the blue team will be able to see the classification of the traffic in real-time and analyze the confidence of the predictions, as well as the IPs and ports that are being classified as cryptomining attacks. The Kibana dashboard also displays the average confidence of the predictions, which the blue team will have to analyze along with the number of attacks detected to determine the best model of the three. Once the blue team has decided which is the most effective model, they will be provided with a command to compare the hash of their selected model against the truly best one in order to determine if their answer is correct. The blue team will also be tasked with providing the IPs of the flows that the selected model has classified as cryptomining attacks. Finally, the blue team is asked to reflect on the differences and advantages between this method and Snort. In particular, the blue team is expected to conclude that Snort will have trouble detecting cryptomining attacks when traffic is encrypted, a situation in which the use of rules for attack detection is completely ineffective. However, since the ML models integrated into the STA component are trained to detect these cryptomining attacks without having to rely on the payload as they are based on flow statistics, the use of ML models can overcome Snort's limitations in this regard. Furthermore, the trainee has to conclude that ML performance is not guaranteed and, therefore, appropriately trained ML models should be used to obtain the best performance.

9.2. DoH Flooding Attack Detection Scenario

In this section, we describe the DoH flooding attack detection scenario that occurs in the data plane of a 5G network. First, we provide an explanation of the DoH flooding attack scenario. We then introduce the integration of ML/DL models into the training scenario for this particular attack. Finally, we describe the proposed exercise to train cybersecurity operators in the use of traditional and more advanced ML/DL-based methods to detect DoH flooding attacks in a 5G network.

9.2.1. DoH Flooding Attack Description

DoH is a DNS query transport protocol that has become widespread in recent years. Its purpose is to allow DNS queries to be sent and received over an encrypted connection, thus ensuring privacy and protection against attacks such as DNS spoofing or eavesdropping. The standard DNS protocol transmits information in plain text and, therefore, does not offer any sort of protection against malicious actors. Using the DoH protocol, the communication between the DoH client and the DoH server is protected by the HTTPS protocol, which uses the TLS/SSL security protocol to encrypt information. In this way, DNS queries remain private and, therefore, the domain name of the websites accessed by users is hidden from the network infrastructure owner.

Attackers can abuse DoH to launch a DoH flooding attack. DoH flooding is a DNS-based denial-of-service (DoS) attack that consists of flooding a DNS server with a large number of DNS queries to overload its network bandwidth and computational resources in order to prevent it from processing legitimate queries [44]. For this purpose, the attacker will send many DNS queries to the DNS server. One of the main features introduced by the DoH protocol is that users can bypass local DNS resolvers and instead query a DoH server directly, thus completely hiding queries from network operators. For this reason, DoH traffic will appear as regular encrypted traffic on the network. Therefore, this malicious traffic sent by the attacker will not be able to be filtered on its way through the network and will inevitably reach the DNS resolver. The DNS server will be unable to process all queries it receives due to its high number, leading to a DNS server DoS. It is important to note that the DNS flooding attack should not be confused with a DNS tunneling or DNS data exfiltration attack because, although they also rely on sending a large number of DNS queries to carry out the malicious activity, their purpose is not to exhaust the DNS server resources, but rather to exfiltrate data from the network.

Despite its simple nature, this attack can cause significant overload on DNS servers and can result in complete denial of service for DNS servers, making the 5G network unavailable for legitimate users to perform this task. For this reason, it is important for cybersecurity operators to be able to detect signs of this attack on a 5G network. Since DoH is an encrypted communication protocol, detecting this attack will require more advanced methods, such as the use of ML/DL models, as traditional rule-based NIDS, such as Snort, are known to be ineffective and impractical in this context. Moreover, even for unencrypted DNS flooding attacks, these traditional defensive methods are known to be very inflexible and can be easily evaded. For this reason, traditional methods cannot be relied upon when dealing with critical infrastructure, and more advanced methods are required. Therefore, to address this critical issue, in this exercise, we propose training cybersecurity operators in the use of these more advanced ML/DL-based methods to successfully deal with this threat and demonstrate their advantages over classical methods.

9.2.2. DoH Flooding Attack Detection Using Machine Learning

To effectively protect against DNS flooding attacks, it is crucial to prevent this malicious traffic from reaching the targeted DNS servers. For this reason, detection and mitigation of this type of attack are typically performed at intermediate nodes in the network by analyzing DNS traffic (i.e., DNS queries and responses) in transit to and from DNS servers. DNS flooding attacks can be detected and filtered at the network level with a variety of methods based on the statistical analysis of DNS packets. This can be achieved, for example, by blacklisting domains or source IPs or by analyzing the frequency of DNS resolution performed by each client and dropping its packets if abnormal behavior is observed [44,45]. However, since DoH encapsulates DNS traffic using the HTTPS secure protocol, DoH traffic is not visible to the network infrastructure, being only available at the DoH client and server. This makes traditional DNS flooding attack detection methods that rely on deep inspection of DNS packets completely obsolete [45]. Furthermore, attackers can use the HTTP/2 connection (which is the minimum version of the HTTP protocol that the latest DoH standard defined in RFC8484 recommends for DoH use [46]) to send multiple DoH requests without creating a separate connection for each request (i.e., multiple packets are sent in the same request). The same is true for the responses that the DoH server returns to the client. Through this method, malware can hide the frequency of its DNS resolution, further reducing the number of methods available to detect DNS flooding [45]. Attackers can exploit these vulnerabilities in the DoH protocol to evade the security mechanisms currently implemented for the detection of flooding attacks. Despite the advantages of DNS over HTTPS (DoH) in terms of user privacy and security, it is clear that its current specification has critical weaknesses that can be exploited to cause considerable harm to the network infrastructure while increasing the difficulty for network administrators to deal with these attacks in order to prevent the misuse of network resources.

To the best of our knowledge, there are no works that address the specific problem of detecting DNS flooding attacks based on the DoH protocol. Other related approaches proposed in the literature have focused mainly on protection against tunneling attacks, such as [45,47], but, as we have pointed out above, the problem of DoH flooding attacks has been neglected so far in the scientific community. The adoption of the DoH protocol is on the rise, and DNS flooding attacks using this protocol are very likely to increase in frequency in the near future. Therefore, the development of new mechanisms to detect DNS flooding attacks is essential to prevent the misuse of network resources and the denial of DNS services to end users.

To fill this gap, we propose a novel technique for the real-time detection of network-level DoH flooding attacks based on ML models that use a large set of network flow statistics as features. Using these features, we were able to train several ML models with different performance levels that were later integrated as part of the DoH STA toolbox. This DoH STA toolbox could then be used during the proposed exercise on the SPIDER cyber range to help the trainee defend against DoH flooding attacks.

9.2.3. Application and Integration of Machine Learning for DoH Flooding Attack Detection

The preprocessing of the data for the DoH scenario was very similar to the one in the cryptomining scenario described in Section 9.1.2. One of the main differences was the tagging of the traffic connections included in the dataset, which now need to be separated into the following classes: 0 (Normal traffic), 1 (Standard DNS over HTTPS traffic), and 2 (DoH flooding attack traffic). For the training of this model, we decided to explore the first 37 features that Tstat statistics provide. In contrast with the cryptomining exercise, in this exercise, we selected a larger number of features to warn the trainee that different feature selection strategies can be applied to ML models. After eliminating the four features containing information about the ports and IPs, which cannot be used as training data as they were used to label the traffic connections, and two features that contained timestamps and that did not only offer any additional information but biased our results, we used the remaining 31 features for the training of our models. We list these features in Table 4. Note that some features are sometimes separated into Client to Server (CS) and Server to Client (SC) communication. These features appear in the table with different CS and SC identifiers.

Table 4. Tstat selected features for DoH flooding attack detection.

CS	SC	Name	Metric	Description
3	17	packets	-	Total number of packets observed from the client/server
4	18	RST sent	0/1	0 = no RST segment has been sent by the client/server
5	19	ACK sent	-	Number of segments with the ACK field set to 1
6	20	PURE ACK sent	-	Number of segments with ACK field set to 1 and no data
7	21	unique bytes	bytes	Number of bytes sent in the payload
8	22	data pkts	-	Number of segments with payload
9	23	data bytes	bytes	Number of bytes transmitted in the payload, including retransmissions
10	24	rexmit pkts	-	Number of retransmitted segments
11	25	rexmit bytes	bytes	Number of retransmitted bytes
12	26	out seq pkts	-	Number of segments observed out of sequence
13	27	SYN count	-	Number of SYN segments observed (including rtx)
14	28	FIN count	-	Number of FIN segments observed (including rtx)
31		Completion time	ms	Flow duration since first packet to last packet
32		C first payload	ms	Client first segment with payload since the first flow segment
33		S first payload	ms	Server first segment with payload since the first flow segment
34		C last payload	ms	Client last segment with payload since the first flow segment
35		S last payload	ms	Server last segment with payload since the first flow segment
36		C first ack	-	Client first ACK segment (without SYN) since the first flow segment
37		S first ack	-	Server first ACK segment (without SYN) since the first flow segment

In the same way as the previous exercise, we decided to create three different models with varying levels of precision, which could teach the blue team that ML-based tools are not fool-proof and can sometimes offer wrong results if trained incorrectly, regardless of the problem being resolved. The hyperparameters with which we decided to experiment were also the number of estimators and the maximum depth, which are both explained in Section 9.1.2. However, a key difference in this exercise is that the third model, named 'doh_rf_31_c.pkl', was trained, including only normal traffic, to force it to always predict all traffic as belonging to that particular class. In that way, the users will be provided with a model that is useless for identifying DoH traffic and which they can use in one of the sections to poison a correct model. To all the models, a random selection of values for the aforementioned hyperparameters was performed to obtain a variety of models with different performance levels. We list these values in Table 5.

Table 5. Hyperparameters for each DoH Random Forest Classifier model generated.

Model Name	Max. Depth	Num. Estimators
doh_rf_31_a.pkl (Good performance)	Unlimited	100
doh_rf_31_b.pkl (Medium performance)	8	3
doh_rf_31_c.pkl (Bad performance)	2	1

An additional challenge for the blue team in this exercise is to test models created with a different ML algorithm. For this purpose, we created two models using Fully Connected Neural Networks (FCNN). Similarly to the other models, we trained them with a different number of layers to yield varying degrees of performance. The structure of each of the FCNN models is specified in Table 6. The model structures are described using a compact notation that is common in the ML literature [48]. This notation is described below. In the first line, a list of values describes the FCNN architecture: $[X_0, X_1, \ldots, X_i, \ldots, X_n]$ where n is the number of fully connected layers and X_i is the number of units in the fully connected layer i. In addition. The activation function of each fully connected layer is listed below. Finally, the final activation function that serves as the output of the model is listed at the bottom. The training hyperparameters for the two models used in the DoH exercise were manually tuned to achieve the intended performance. The training hyperparameters of both models are the same, except for the maximum number of epochs, the value of which is given in Table 6 for each model. In particular, we use a batch size of 2048. We also use the Adam optimizer with a learning rate of 0.001. In addition, we use the early stopping technique to automatically terminate the training process if the validation loss does not improve for two epochs. For the validation procedure, we reserve 10% of the training data for the validation split. Finally, as a loss function, we use the categorical cross-entropy function.

The quality measures of Table 7 display how the variation in the hyperparameters and structure affects the model's precision and the generation of false positives and false negatives. The quality measures that we have taken into account are the same as the ones we used to evaluate the cryptomining attack detection models: F1-Score, balanced accuracy, and confusion matrix. We also used balanced accuracy in this case, to evaluate the models, as the DoH dataset is not perfectly balanced.

Table 6. Hyperparameters for each DoH Deep Learning model generated.

Model Name	Structure	Max. Epochs	Batch Size
doh_dl_a.pkl (Good performance)	FC Layers: [62, 31, 15, 3] Activation: ReLU Final Activation: SoftMax	100	2048
doh_dl_b.pkl (Medium performance)	FC Layers: [62] Activation: ReLU Final Activation: SoftMax	1	2048

FC: Fully Connected Layer.

Table 7. Quality measures of DoH traffic classification models. In the confusion matrices 0: NORMAL, 1: DOH, 2: DDOS DOH, where rows represent true values and columns predicted values.

Model Name	Quality Measure	Score			
doh_rf_31_a.pkl (Good performance)	F1-Score	0.940			
	Balanced Accuracy	0.935			
	Confusion Matrix		**0**	**1**	**2**
		0	199,645	2174	790
		1	6977	116,008	12,343
		2	6257	246	169,298
doh_rf_31_b.pkl (Medium performance)	F1-Score	0.620			
	Balanced Accuracy	0.654			
	Confusion Matrix		**0**	**1**	**2**
		0	193,883	0	8726
		1	34,377	24,348	76,603
		2	30,824	0	144,977
doh_rf_31_c.pkl (Bad performance)	F1-Score	0.189			
	Balanced Accuracy	0.333			
	Confusion Matrix		**0**	**1**	**2**
		0	202,609	0	0
		1	135,328	0	0
		2	175,801	0	0
doh_dl_31_a.pkl (Good performance)	F1-Score	0.750			
	Balanced Accuracy	0.759			
	Confusion Matrix		**0**	**1**	**2**
		0	194,575	2165	5869
		1	7220	48,059	80,049
		2	6220	521	169,060
doh_dl_31_b.pkl (Medium performance)	F1-Score	0.665			
	Balanced Accuracy	0.702			
	Confusion Matrix		**0**	**1**	**2**
		0	194,700	1643	6266
		1	8525	24,447	102,356
		2	6166	27	169,608

In the same way as in the cryptomining exercise, by observing Table 7, we can clearly appreciate the didactic value of this exercise. By applying ML models to detect DoH traffic, the blue team can test real traffic against different models observing the contrasting results, varying in confidence and consistency of detecting attacks. This inconsistency in detecting attacks will be clearly observed in the third model, 'doh_rf_31_c.pkl', which will show the blue team that a model can also be modified to always predict the same result, something that they will be able to apply when taking the role of the attacker to poison the model. In the case of our exercise, the model poisoning method consists of obtaining root access to the machine and overwriting the best model with the bad model mentioned above. Although there are other ways of poisoning an ML model, we decided to use this method because it is considered one of the most straightforward methods to accomplish this task, while also being highly effective. More sophisticated methods of model poisoning that are more difficult to detect or do not require privilege escalation could also be applied at the cost of increased complexity, but they are beyond the scope of our exercise, as we only intend to provide a potential vulnerability that could be effectively exploited to compromise the model and its results to teach the blue team to be aware of this type of attack vector when working with ML models. By including models with different ML technologies to classify traffic, we show the blue team that the field is diverse and several different methods, with varying degrees of performance and efficiency, may be applied according to the particular needs of the organization. In conclusion, similar to the cryptomining exercise, this exercise provides a scenario that perfectly mimics a real-life situation, tasking the trainee to evaluate different ML models to compare their performance and allowing the end user to judge which one is best suited to their particular needs. In this way, this exercise aims to reveal the potential of this technology and its possible pitfalls when applied incorrectly.

9.2.4. Description of the DoH Flooding Attack Exercise

The scenario consists of two virtual machines communicating with each other. The first VM uses the tcpreplay tool to replay real network traffic from various PCAP files. Network traffic is generated and stored in PCAP files in the Machine Learning Lab for this exercise and includes different 5G DNS infrastructure attacks captured in the UPF plane over the GTP protocol. The traffic involved includes multiple 5G user sessions, flooding attacks against 5G DNSo53 (DNS over UDP on port 53), and DoH (DNS over HTTPS). The second VM includes the tools to analyze and detect attacks. The exercise emulates a situation where the malicious DNS attack must be detected before leaving the UPF node on the N9 interface for communication between UPF nodes or in a roaming scenario. This scenario is illustrated in Figure 11.

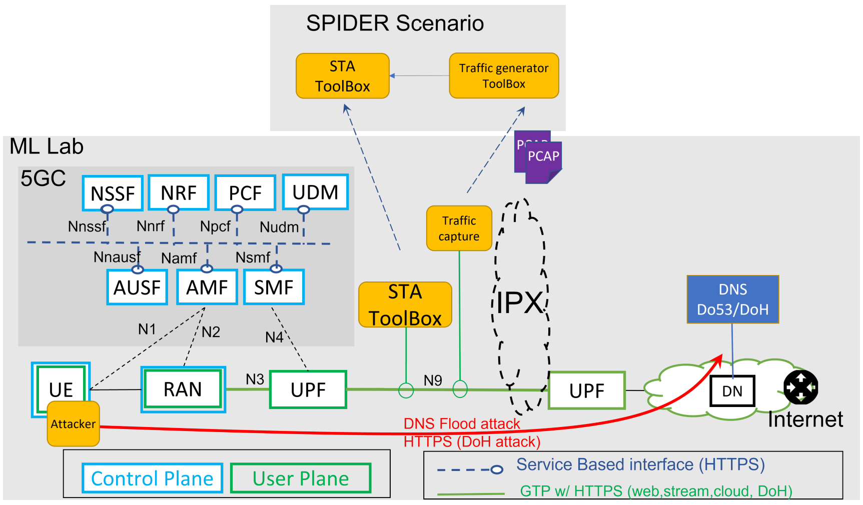

Figure 11. DNS attack scenario.

All the phases and processes the blue team needs to perform to complete the DoH flooding attack detection exercise are described in Figure 12. In the following paragraphs, we describe, in detail, the different phases shown in the diagram.

Starting the exercise with an introduction in Phase A, the blue team is asked to assume the role of a security expert in the SOC (Security Operative Center) of a 5G cloud infrastructure provider. In this role, the blue team receives a ticket from the 5G operator team, reporting that they are experiencing performance problems in their 5G IPX access. The blue team must hence analyze the traffic provided to them, detect the compromised devices, and identify the malware in the form of DNS and DoH flooding attacks that are impacting the performance. In order to achieve this, the blue team is asked to follow steps that include different detection strategies, including Snort analysis and a novel ML-based tool with several inference engines customized for different attacks.

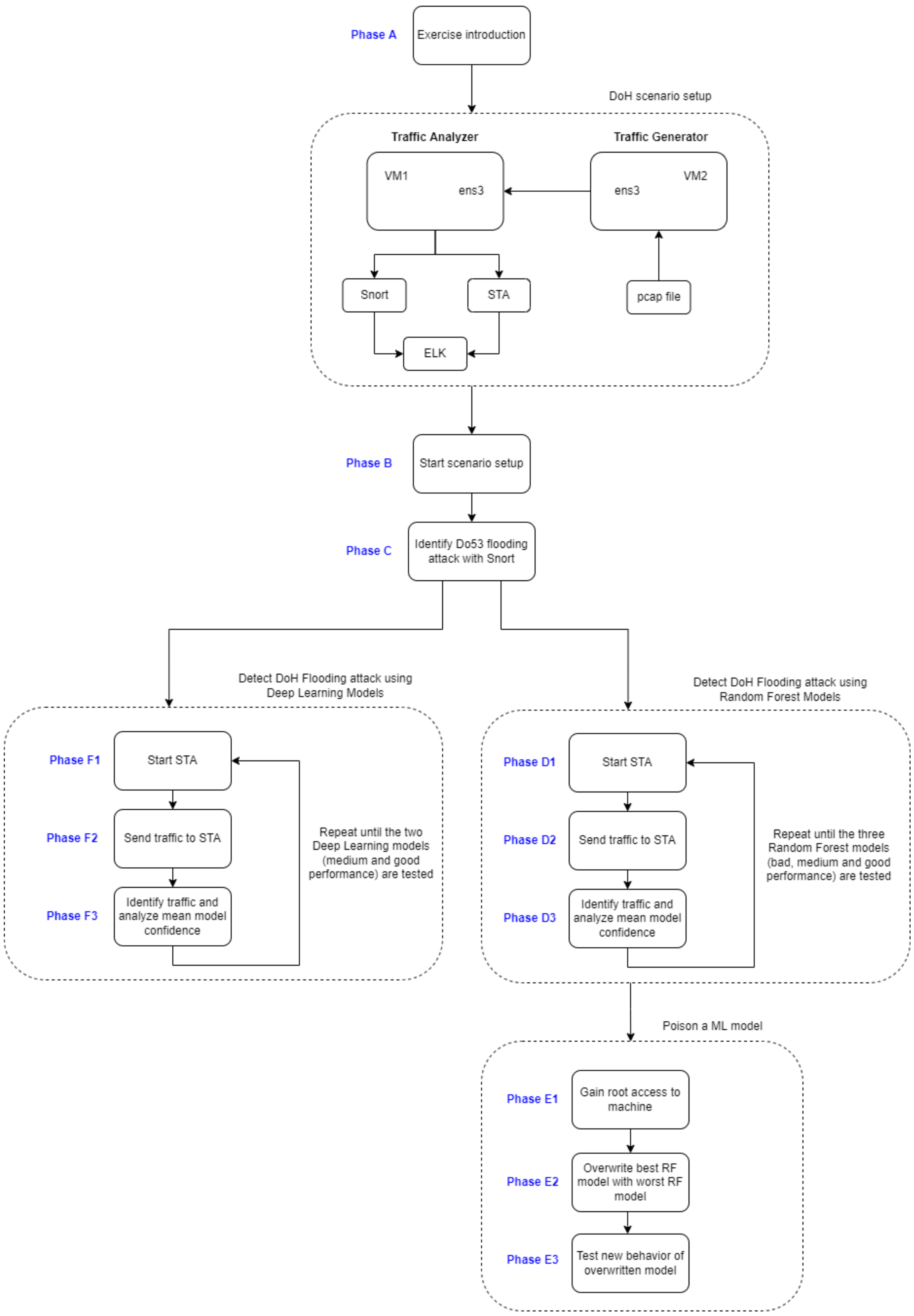
Phase A
Exercise introduction
DoH scenario setup
Traffic Analyzer
Traffic Generator
VM1
ens3
ens3
VM2
Snort
STA
pcap file
ELK
Phase B
Start scenario setup
Phase C
Identify Do53 flooding attack with Snort
Detect DoH Flooding attack using Deep Learning Models
Detect DoH Flooding attack using Random Forest Models
Phase F1
Start STA
Phase F2
Send traffic to STA
Phase F3
Identify traffic and analyze mean model confidence
Repeat until the two Deep Learning models (medium and good performance) are tested
Phase D1
Start STA
Phase D2
Send traffic to STA
Phase D3
Identify traffic and analyze mean model confidence
Repeat until the three Random Forest models (bad, medium and good performance) are tested
Poison a ML model
Phase E1
Gain root access to machine
Phase E2
Overwrite best RF model with worst RF model
Phase E3
Test new behavior of overwritten model

Figure 12. DoH scenario phases.

Part 1: Detecting Classical DNS Flooding Attack with Snort

In this section, which corresponds to Phase C, the blue team will be tasked with using the Snort tool on the first machine, which we will refer to as VM1. The tool is available in a container that the blue team will have to start using a provided script. Once the container is active, the blue team will be able to inject the traffic into it. To do this, the blue team machine will run another shell script file already prepared on the traffic generator machine, which from now on, we will call VM2. Back in the Kibana dashboard, the blue team is tasked with observing the alerts generated by Snort. However, by observing the information displayed, they will eventually find out that Snort is not showing any alerts since the rules are intentionally not correctly written in order to teach them to specify the correct rules to detect classical DNS flooding attack (DNS over UDP, port 53—Do53 for short) traffic connections. Using several valuable resources on the subject provided along with the exercise, the blue team will have to update the existing rules so that they will correctly send alerts to Elasticsearch. After researching the resources provided in the instructions, the blue team will be able to edit a specific configuration file and rewrite the rules to detect the Do53 flooding attacks. The blue team will then be able to test their rules by restarting the Snort container and observing the alerts displayed on the Kibana dashboard. Using the results, the blue team is then asked to provide the IPs that have been identified as Snort for belonging to Do53 flooding attacks and the servers that are being compromised. The blue team is then asked to reflect on the effectiveness of Snort in analyzing DoH traffic instead of unencrypted Do53 traffic and should conclude that this tool will have difficulty detecting attacks that are being performed using the DoH protocol due to the encryption. To address this issue, in the next part, we introduce the use of the Smart Traffic Analyzer (STA) that will enable the detection of DoH flooding attacks. As can be observed in Figure 12, this phase branches out into two different phases. This is meant to show that even though they are presented in sequential order in the exercise, these phases are independent of one another and can be performed in any order. Completing these phases in a different order will not have an impact on the final score of the blue team due to the nature of the evaluation process, reflected in Section 9.3.

Part 2: Detecting DoH Flooding Attack with STA

In this next step of the exercise, the blue team will be provided with three different ML models trained with the same dataset to detect DoH flooding attacks. As was the case in the cryptomining attack detection exercise, the difference between these three models is, once again, the hyperparameters used to train them, which will influence the confidence of the predictions and, ultimately, the number of attacks detected by each model. Furthermore, the model named 'doh_rf_31_c.pkl' has only been trained with normal traffic so that it predicts everything belonging to this particular class. As part of Phase D1, the blue team will begin the section by starting the STA containers in the VM1 with the provided command and restarting it with a different model each time. Then, the blue team will follow Phase D2, where they will send traffic from VM2 using the methods described in the previous sections; the STA container will be able to analyze it and classify it according to the predicted class. In Phase D3, the blue team will access the Kibana display to be able to see the classification of the traffic in real-time and perform an analysis on the confidence of the predictions, as well as the IPs and ports which are being classified as DoH flooding attacks. To facilitate the examination of the results, the Kibana dashboard also displays the average confidence of the predictions, which the blue team will have to take into account, along with the number of attacks detected to determine the best model. The blue team will be provided a command to compare the hash of their selected model against the best one in order to determine if their answer is correct. Lastly, the blue team is asked to reflect on the difference between this method and the previously mentioned Snort. They are expected to conclude that the STA's ability to discern attacks, regardless of encryption, is a key advantage over the use of Snort.

Part 3: Poisoning a Machine Learning Model

The purpose of this part is for the blue team to act as an attacker whose goal is to poison the ML model in order to manipulate its behavior in such a way that it completely inhibits its detection capability and, therefore, does not notify the operator of DoH flooding attacks that may be occurring in the network. The attacker is tasked with reading documentation about the LinPEAS script [49], which is a tool that detects vulnerabilities in a system and uses this information to gain root access to the machine. This step is labeled Phase E1. With this root access, the attacker will transition to Phase E2, where they must replace the best model with the poisoned one, which classifies all traffic as normal, as previously mentioned in the above section. Once this change has been made, in Phase E3, the blue team is asked to start the STA container again using the model that was previously the best and verify that all traffic is being classified as normal. The blue team will then have to execute a command to test the hash of the best model file against the malicious one to ensure that they have completed the exercise correctly. The blue team is then asked to reflect on how this type of exploitation could have been prevented. A complementary goal of this section is for the blue team to reach the conclusion that cybersecurity must be taken into account in every step of the process, and a vulnerable environment lends itself to multiple types of attacks. This lesson, therefore, encourages security teams to be proactive in the search for vulnerabilities that could be exploited to enable this kind of attack, which, as this exercise has shown, could result in unfortunate consequences if not properly mitigated. In this case, the vulnerability could have been spotted by the blue team using the same script that the attacker used and could have been fixed to prevent this type of privilege escalation.

Part 4: Using a Different ML Technology

Before starting this last section, the blue team is asked to execute a script that is already created to reset the models to their initial state, thus clearing the model poisoning performed in the previous step. This step is not explicitly needed if the blue team has saved the results of their analysis on the three Random Forest models previously tested, but it will be useful if they need to use those models again for testing purposes. Starting in Phase F1, the blue team will be tasked with restarting the STA containers in the VM1 with the provided command and testing the new models that are based on Neural Networks instead of the Random Forest models that have been used so far. In the same way as the previous sections, the blue team will send traffic from VM2 in Phase F2 and will analyze the results shown in Kibana in Phase F3. The blue team is encouraged to compare the mean confidence and DoH flooding attacks detected by each model. This section is meant to introduce the blue team to a different ML technology (Neural Networks, in this case) and show them how the same problem may be resolved effectively using different tools. This is a valuable lesson, as the use of a different ML algorithm to build the detection models may provide some advantages in terms of performance, efficiency, or easier training and, thus, is an important aspect to take into account when implementing detection solutions that are based on ML technology. Finally, the blue team will be asked to provide the IPs of the attackers and the IPs of the compromised servers that they have detected.

9.3. Performance Evaluation of the Exercises Executed by the Blue Team

The Evaluation Agent is in charge of monitoring the commands executed by the trainee host during the exercise and evaluating the performance of the blue team based on the correct answers at the end of the session. To do this, the Evaluation Agent records the commands executed by the blue team and their corresponding output during the exercise. The output of the script command is saved in the file `/tmp/exercise\_<crypto|doh>.out`. The file is transferred at the end of the session to the Evaluation Engine, which uses this information to generate the performance report. The file transfer is performed with an HTTP POST request to the Evaluation Engine. The HTTP server running in the Evaluation Engine waits until it receives the file and stores it in the `/tmp/` folder. The Evaluation

Engine then compares the commands executed by the blue computer with the expected commands in the exercise. The output of the expected commands is stored in `/opt/exercise\_<crypto|doh>.sol`. The Evaluation Engine determines the blue team's score based on the matches between the output of the executed commands and the expected output. Each correct command is only considered once to avoid alteration of the results. The final score of the exercise is calculated by summing the number of matches.

Once the analysis is complete, the Evaluation Engine provides the blue team's score to the Evaluation Agent by sending an HTTP GET request to the Evaluation Agent. The Evaluation Agent then sends this information to the operational dashboard to inform the blue team of the evaluation results. To send this information, the Evaluation Agent uses a Kafka producer to send the information to the operational dashboard. The operational dashboard subscribes to the Kafka topic to receive the information. When the information is received, the operational dashboard prints this information on the console and sends a notification to the blue team. The operational dashboard is also responsible for storing the information in the database so that it can be consulted later. For convenience, the performance results are also printed on the terminal where the exercise has been performed.

10. Conclusions, Open Challenges, and Future Work

In this section, we summarize and conclude the main findings of this work. Additionally, we briefly discuss several open challenges to be considered that are interesting, and finally, we suggest interesting future work to explore.

10.1. Conclusions

This work presented the integration of machine learning-based attack detection exercises into SPIDER, an innovative Cyber Range as a Service (CRaaS) for 5G networks. Two recently appeared attacks, cryptomining attack and DNS over HTTP (DoH) flooding, were selected as support use cases for the proposed exercises. For didactic purposes, poisoning techniques were also introduced in the DoH flooding exercise. The proposed exercises use realistic 5G network traffic that is generated in a new environment based on a fully virtualized 5G network. We can conclude in this respect that in addition to the usefulness of experimenting with the detection of two recently appeared attacks, the proposed exercises help to exemplify how other attacks and their counterparts, the ML-based attack detectors, can be integrated as cyber range exercises.

From an industrial perspective, the proposed Smart Traffic Analyzer (STA) using container technology and standard interfaces will facilitate the design and deployment of new exercises based on the emulation of other attacks.

Finally, the application of the recently discovered Generative Adversarial Networks (GANs) to obtain on-demand synthetic flow-based network traffic allows replication of high fidelity network attacks and normal traffic and, therefore, the synthetic data can be seamlessly integrated into SPIDER exercises to be used instead of real traffic. This solution allows for the generation of unlimited amounts of synthetic data for a concrete type of exercise and thus prevents trainees from always learning an exercise with the same set of real data. In addition, this GAN-based solution can foster the interconnection of federations of cyber ranges as it allows importing and exporting network traffic data from one cyber range to another without incurring any privacy violation as the shared data are only synthetic.

10.2. Open Challenges

Although the development and deployment of a cyber range solution is a very valuable contribution to the security community and has proven to be a powerful tool for assessing the security capabilities of systems and networks, the applicability of this solution to real-world scenarios is still limited. In this subsection, we identify some open challenges related to our work that are worth addressing to increase the applicability of the proposed solution to real-world scenarios.

1. **Lack of contextualization:** Although ML models can be very effective in characterizing and detecting malicious behavior on the network, they often lack the ability to take advantage of contextual information to provide the user with a clear understanding of the events that have been detected and the potential impact of the threat. In addition, ML models do not take into account organization-specific policies, which limits the operator's ability to move beyond the detection of an attack event to achieve end-to-end protection of the system. ML models are not sufficient to fully automate the process of detecting and mitigating an attack, and considerable effort is still required on the part of the operator to understand and react appropriately to the detected threat.

As long as cyber ranges do not have that capability and do not teach the trainee how to take advantage of all available contextual information to detect attacks efficiently and implement mitigations, the usefulness of Ml-based exercises to detect and mitigate threats in real-world settings is limited.

2. **Difficulty of data collection and labeling:** One of the most difficult tasks that organizations are often confronted with in the design and development of a cyber range solution is the collection and labeling of a realistic dataset that can be used to train and evaluate the performance of ML models. The lack of public datasets that cover a wide range of different types of attacks (e.g., network intrusion detection, malware detection, data exfiltration, etc.) is one of the main limitations to the development of an efficient cyber range system. This situation has forced organizations to gather their own datasets. However, this brings with it a number of technical challenges that organizations often have trouble justifying. In the case of cybersecurity applications, collecting a realistic dataset is a highly challenging task due to the many different types of attacks that can be carried out and the large amount of data that must be collected in order to correctly model the intrinsic variability of traffic data, which makes the time required to tag each data sample an arduous task that requires a team of experts. Collecting and labeling a realistic dataset is compounded by the possibility of new attacks appearing that are not yet well characterized. While collecting network traffic is a relatively straightforward task that can be completed through the use of a passive monitoring system, labeling data that have been collected "in the wild" is not possible, as the traffic is often encrypted. Therefore, simulation tools are required to replicate these types of attacks in a controlled environment where labels can be added as the network traffic is generated in a controlled way. However, due to the increasing complexity of network traffic, simulation environments are often limited in size and not realistic enough, which can lead to the development of ML models that lack real-world effectiveness due to the oversimplification of attack scenarios.

In this context, the Mouseworld Lab proposes a controlled and realistic scenario where normal and attack connections can be emulated in a realistic 5G network. Furthermore, the generated packets can be collected and aggregated into connection statistics, and the datasets containing these connections can be labeled automatically. However, one of the topics currently being researched in the Mouseworld Lab is how to set up new attack emulations in an automated way, as nowadays, every attack scenario needs to be configured manually to a large extent.

10.3. Future Work

In future work, we propose to develop offensive attack exercises supported by ML models for the red team. In these exercises, our GAN-based solution could be applied to generate variants of a specific attack and penetration tests.

Adversarial examples are a topic of great interest in the area of information security and defense against cybersecurity threats. In this context, we propose the development of exercises for the red team that contains adversarial examples that can fool attack detection models while appearing as normal traffic to human analysts. Additionally, we propose creating lessons and exercises that teach the blue team how to defend against these adversarial attacks to increase the robustness and resilience of ML-based detection systems.

This work focused on the integration of ML-based detectors in defensive exercises, but the integration in these exercises of mitigation strategies to be applied after a successful detection of an attack is another interesting research line that will be explored in the future. This line of research is not trivial as it will involve additional synchronization between the SPIDER cyber range running the exercises and the Mouseworld Lab acting as a network digital twin that not only emulates the attacks but also receives the mitigation actions from the trainee and applies them in real-time in the emulated scenario, which will produce changes not initially programmed in the emulation.

Defensive exercises based on unsupervised ML techniques can also be a perfect complement to the exercises developed in this work, as they allow the trainee to learn in which attack scenarios it is better to apply unsupervised or supervised techniques or a combination of both. The integration of unsupervised ML techniques into SPIDER exercises must address the design of new messages between the STA and the SPIDER components that interact with it (e.g., the Dashboard and the Evaluation Agent) as the output structure generated by unsupervised ML models is quite different from that of supervised methods.

Author Contributions: Conceptualization, A.M., A.P., A.K., L.d.l.C., J.I.M. and D.R.; methodology, A.M., A.P., A.K., L.d.l.C., J.I.M. and D.R.; software, A.M., A.P., A.K., L.d.l.C., J.I.M. and D.R.; validation, A.M., A.P., A.K., L.d.l.C., J.I.M. and D.R.; investigation, A.M., A.P., A.K., L.d.l.C., J.I.M. and D.R.; data curation, A.M., A.P., A.K., L.d.l.C., J.I.M. and D.R.; writing—original draft preparation, A.M., A.P., A.K., L.d.l.C., J.I.M. and D.R.; writing—review and editing, A.M., A.P., A.K., L.d.l.C., J.I.M. and D.R.; funding acquisition, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research leading to these results received partial funding from the European Union's Horizon 2020 research and innovation programme under grant agreement nº833685 (SPIDER) and by the Spanish Ministerio de Asuntos Económicos y Transformación Digital, Programa UNICO under project B5GEMINI (TSI-063000-2021-79, TSI-063000-2021-81).

Acknowledgments: We thank Hugo Ramón Pascual (Telefónica I+D) for his valuable help in the design, implementation, integration, and deployment of the cryptomining detection and DoH flooding attack exercises. We thank Panagiotis Gouvas (UBITECH), technical leader of the SPIDER project, and the other SPIDER collaborators, who contributed their knowledge and experience in fruitful discussions on the SPIDER architecture during the realization of the project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sultana, N.; Chilamkurti, N.; Peng, W.; Alhadad, R. Survey on SDN Based Network Intrusion Detection System Using Machine Learning Approaches. *Peer-Peer Netw. Appl.* **2019**, *12*, 493–501. [CrossRef]
2. Hu, W.; Tan, Y. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. *arXiv* **2017**, arXiv:1702.05983. https://doi.org/10.48550/arXiv.1702.05983.
3. Pastor, A.; Mozo, A.; Lopez, D.R.; Folgueira, J.; Kapodistria, A. The Mouseworld, a security traffic analysis lab based on NFV/SDN. In Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, 27–30 August 2018; pp. 1–6.
4. Cohen, F. Simulating Cyber Attacks, Defences, and Consequences. *Comput. Secur.* **1999**, *18*, 479–518. [CrossRef]
5. Mirkovic, J.; Benzel, T. Teaching Cybersecurity with DeterLab. *IEEE Secur. Priv.* **2012**, *10*, 73–76. [CrossRef]
6. Turčaník, M. A Cyber Range for Armed Forces Education. *Inf. Secur. Int. J.* **2020**, *46*, 304–310. [CrossRef]
7. Weiss, R.; Turbak, F.; Mache, J.; Locasto, M.E. Cybersecurity Education and Assessment in EDURange. *IEEE Secur. Priv.* **2017**, *15*, 90–95. [CrossRef]
8. Smyrlis, M.; Somarakis, I.; Spanoudakis, G.; Hatzivasilis, G.; Ioannidis, S. CYRA: A Model-Driven CYber Range Assurance Platform. *Appl. Sci.* **2021**, *11*, 5165. [CrossRef]
9. Vykopal, J.; Ošlejšek, R.; Čeleda, P.; Vizvary, M.; Tovarňák, D. Kypo cyber range: Design and use cases. In Proceedings of the 12th International Conference on Software Technologies—ICSOFT, Madrid, Spain, 24–26 July 2017.
10. Pham, C.; Tang, D.; Chinen, K.i.; Beuran, R. CyRIS: A Cyber Range Instantiation System for Facilitating Security Training. In Proceedings of the Seventh Symposium on Information and Communication Technology, SoICT '16, Ho Chi Minh City, Vietnam, 8–9 December 2016; pp. 251–258. [CrossRef]

11. Yamin, M.M.; Katt, B.; Gkioulos, V. Cyber Ranges and Security Testbeds: Scenarios, Functions, Tools and Architecture. *Comput. Secur.* **2020**, *88*, 101636. [CrossRef]
12. Ukwandu, E.; Farah, M.A.B.; Hindy, H.; Brosset, D.; Kavallieros, D.; Atkinson, R.; Tachtatzis, C.; Bures, M.; Andonovic, I.; Bellekens, X. A Review of Cyber-Ranges and Test-Beds: Current and Future Trends. *arXiv* **2020**, arXiv:2010.06850.
13. Chouliaras, N.; Kittes, G.; Kantzavelou, I.; Maglaras, L.; Pantziou, G.; Ferrag, M.A. Cyber Ranges and TestBeds for Education, Training, and Research. *Appl. Sci.* **2021**, *11*, 1809. [CrossRef]
14. Costa, G.; Russo, E.; Armando, A. Automating the Generation of Cyber Range Virtual Scenarios with VSDL. *arXiv* **2020**, arXiv:2001.06681.
15. Gustafsson, T.; Almroth, J. Cyber Range Automation Overview with a Case Study of CRATE. In Proceedings of the 25th Nordic Conference, NordSec 2020, Virtual Event, 23–24 November 2020; pp. 192–209. [CrossRef]
16. Brilingaitė, A.; Bukauskas, L.; Juozapavičius, A. A Framework for Competence Development and Assessment in Hybrid Cybersecurity Exercises. *Comput. Secur.* **2020**, *88*, 101607. [CrossRef]
17. Vykopal, J.; Vizvary, M.; Oslejsek, R.; Celeda, P.; Tovarnak, D. Lessons Learned from Complex Hands-on Defence Exercises in a Cyber Range. In Proceedings of the 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, USA, 18–21 October 2017; pp. 1–8. [CrossRef]
18. Xenakis, C.; Angelogianni, A.; Veroni, E.; Karapistoli, E.; Ghering, M.; Gerosavva, N.; Machamint, V.; Polvanesi, P.; Brignone, A.; Mendoza, J.N.; et al. The SPIDER concept: A Cyber Range as a Service platform. In Proceedings of the European Conference on Networks and Communications (Eucnc2020), Virtual, 16–17 June 2020. [CrossRef]
19. Sommer, R.; Paxson, V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 305–316. [CrossRef]
20. Jirsik, T.; Husák, M.; Celeda, P.; Eichler, Z. Cloud-Based Security Research Testbed: A DDoS Use Case. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; pp. 1–2. [CrossRef]
21. Gavaudan, L.; Legras, S.; Ventos, V. Cyber range automation, a bedrock for AI applications. In Proceedings of the 28th C&ESAR, Rennes, France, 16–17 November 2021; p. 165.
22. Ahmad, I.; Kumar, T.; Liyanage, M.; Okwuibe, J.; Ylianttila, M.; Gurtov, A. Overview of 5G Security Challenges and Solutions. *IEEE Commun. Stand. Mag.* **2018**, *2*, 36–43. [CrossRef]
23. Apruzzese, G.; Colajanni, M.; Ferretti, L.; Guido, A.; Marchetti, M. On the Effectiveness of Machine and Deep Learning for Cyber Security. In Proceedings of the 2018 10th International Conference on Cyber Conflict (CyCon), Tallinn, Estonia, 29 May–1 June 2018; pp. 371–390. [CrossRef]
24. Liu, H.; Han, W.; jia, Y. Construction of Cyber Range Network Security Indication System Based on Deep Learning. In Proceedings of the 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC), Hangzhou, China, 23–25 June 2019; pp. 495–502. [CrossRef]
25. Pitropakis, N.; Panaousis, E.; Giannetsos, T.; Anastasiadis, E.; Loukas, G. A taxonomy and survey of attacks against machine learning. *Comput. Sci. Rev.* **2019**, *34*, 100199. [CrossRef]
26. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial machine learning at scale. *arXiv* **2016**, arXiv:1611.01236.
27. Wiyatno, R.R.; Xu, A.; Dia, O.; de Berker, A. Adversarial examples in modern machine learning: A review. *arXiv* **2019**, arXiv:1911.05268.
28. Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Xu, M. A survey on machine learning techniques for cyber security in the last decade. *IEEE Access* **2020**, *8*, 222310–222354. [CrossRef]
29. Ibitoye, O.; Abou-Khamis, R.; Matrawy, A.; Shafiq, M.O. The Threat of Adversarial Attacks on Machine Learning in Network Security—A Survey. *arXiv* **2019**, arXiv:1911.02621.
30. European Telecommunications Standards Institute (ETSI). *ETSI GR SAI 004 V1.1.1 (2020-12). Securing Artificial Intelligence (SAI); Problem Statement*; ETSI: Sophia Antipolis, France, 2020.
31. Mellia, M.; Carpani, A.; Cigno, R.L. Tstat: TCP statistic and analysis tool. In Proceedings of the International Workshop on Quality of Service in Multiservice IP Networks, Milano, Italy, 24–26 February 2003; pp. 145–157.
32. Claise, B. Cisco Systems NetFlow Services Export Version 9. Request for Comments RFC 3954, Internet Engineering Task Force, 2004. Available online: https://www.rfc-editor.org/info/rfc3954 (accessed on 21 August 2022).
33. ONNX: Open Neural Network Exchange. 2019. Available online: https://onnx.ai (accessed on 20 September 2022).
34. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *arXiv* **2014**, arXiv:1406.2661. https://doi.org/10.48550/arXiv.1406.2661.
35. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 214–223.
36. González-Prieto, Á.; Mozo, A.; Gómez-Canaval, S.; Talavera, E. Improving the quality of generative models through Smirnov transformation. *Inf. Sci.* **2022**, *609*, 1539–1566. [CrossRef]
37. Mozo, A.; González-Prieto, Á.; Pastor, A.; Gómez-Canaval, S.; Talavera, E. Synthetic Flow-Based Cryptomining Attack Generation through Generative Adversarial Networks. *Sci. Rep.* **2022**, *12*, 2091. [CrossRef] [PubMed]
38. González-Prieto, Á.; Mozo, A.; Talavera, E.; Gómez-Canaval, S. Dynamics of fourier modes in torus generative adversarial networks. *Mathematics* **2021**, *9*, 325. [CrossRef]

39. Pastor, A.; Mozo, A.; Vakaruk, S.; Canavese, D.; López, D.R.; Regano, L.; Gómez-Canaval, S.; Lioy, A. Detection of Encrypted Cryptomining Malware Connections With Machine and Deep Learning. *IEEE Access* **2020**, *8*, 158036–158055. [CrossRef]
40. Dini, P.; Saponara, S. Analysis, Design, and Comparison of Machine-Learning Techniques for Networking Intrusion Detection. *Designs* **2021**, *5*, 9. [CrossRef]
41. Zayuelas-Muñoz, J.; Suárez-Varela, J.; Barlet-Ros, P. Detecting Cryptocurrency Miners with NetFlow/IPFIX Network Measurements. In Proceedings of the 2019 IEEE International Symposium on Measurements & Networking (M&N), Catania, Italy, 8–10 July 2019; pp. 1–6. [CrossRef]
42. Swedan, A.; Khuffash, A.N.; Othman, O.; Awad, A. Detection and Prevention of Malicious Cryptocurrency Mining on Internet-Connected Devices. In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, ICFNDS'18, Amman, Jordan, 26–27 June 2018; pp. 1–10. [CrossRef]
43. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
44. Zargar, S.T.; Joshi, J.; Tipper, D. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 2046–2069. [CrossRef]
45. de Vries, L. *Detection of DoH Tunnelling : Comparing Supervised with Unsupervised Learning*; University of Twente: Enschede, The Netherlands, 2021.
46. Hoffman, P.E.; McManus, P. DNS Queries over HTTPS (DoH). Request for Comments RFC 8484, Internet Engineering Task Force, 2018. Available online: https://www.rfc-editor.org/info/rfc8484 (accessed on 10 August 2022).
47. Singh, S.K.; Roy, P.K. Vulnerability Detection of DNS over HTTPS Traffic Using Ensemble Machine Learning. In Proceedings of the 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), Sakhir, Bahrain, 20–21 December 2020; p. 9.
48. Mozo, A.; Segall, I.; Margolin, U.; Gomez-Canaval, S. Scalable prediction of service-level events in datacenter infrastructure using deep neural networks. *IEEE Access* **2019**, *7*, 179779–179798. [CrossRef]
49. Polop, C. LinPEAS—Linux Privilege Escalation Awesome Script. LinPEAS Is a Script that Search for Possible Paths to Escalate Privileges on Linux/Unix*/MacOS Hosts. 2019. Available online: https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS (accessed on 24 September 2022).