Chapter 8
The Role of Machine Learning and Radio Reconfigurability in the Quest for Wireless Security

Francesco Restuccia, Salvatore D'Oro, Liyang Zhang, and Tommaso Melodia

Abstract Wireless networks require fast-acting, effective and efficient security mechanisms able to tackle unpredictable, dynamic, and stealthy attacks. In recent years, we have seen the steadfast rise of technologies based on machine learning and software-defined radios, which provide the necessary tools to address existing and future security threats without the need of direct human-in-the-loop intervention. On the other hand, these techniques have been so far used in an ad hoc fashion, without any tight interaction between the attack detection and mitigation phases. In this chapter, we propose and discuss a Learning-based Wireless Security (LeWiS) framework that provides a closed-loop approach to the problem of cross-layer wireless security. Along with discussing the LeWiS framework, we also survey recent advances in cross-layer wireless security.

8.1 Introduction

Due to the broadcast nature of radio-frequency (RF) waves, wireless networks are particularly vulnerable to a plethora of security threats, including jamming, denial-of-service (DoS), eavesdropping, message falsification/injection, and address spoofing, just to name a few [106, 110, 113, 115]. These threats, when carried out stealthily, may disrupt the network's functionality and seriously compromise users' security and privacy.

Traditionally, wireless attacks have focused on the disruption of a single layer on the network protocol stack by concentrating all the adversary efforts on a single objective. For example, most of the existing jamming techniques focus on disrupting wireless communications by transmitting high-power RF waves on the physical medium [84]. Recently, a number of *cross-layer* wireless attacks [104, 105, 109]

F. Restuccia · S. D'Oro · L. Zhang · T. Melodia (✉)
Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, USA
e-mail: frestuc@northeastern.edu; s.doro@northeastern.edu; liyangzh@ece.neu.edu; melodia@northeastern.edu

C. Wang and Z. Lu (eds.), *Proactive and Dynamic Network Defense*, Advances in Information Security 74, https://doi.org/10.1007/978-3-030-10597-6_8

has been proposed, where *activities* and *objectives* entail different layers of the network protocol stack. The main feature of cross-layer attacks is that they are carried out by attacking layers different than the targeted one (also called *helping layers*). As a consequence, small-scale (and thus, hard-to-detect) attack activities may lead to dramatic changes on the *target layer*. Cross-layer threats are further exacerbated by the fact that many network protocols functionalities such as power allocation, channel selection, and routing decisions are jointly optimized with a common objective [23, 51, 64], resulting in layers that are closely coupled with each other. As long as the helping and target layers are coupled, the attack will lead to the defender's *responsive* change on the target layer. Thus, with carefully-tuned attack activities and objectives, the defender's reaction will favor the attacker's objective.

Cross-layer attacks present unique challenges that cannot be addressed by legacy security techniques. First, cross-layer attacks leverage small-scale activities in the helping layer to achieve significant damage in the target layer. This implies that the attacker can achieve the same goal with relatively small-scale activities, and therefore remain *undetected*. On the other hand, existing attack detection methods often assume that attacks are conducted always in the same manner and always have the same objective, and that large-scale attacks have to be conducted in order to achieve substantial results [110]. This is not necessarily true in cross-layer attacks. Therefore, developing *detection* and *mitigation* algorithms able to swiftly detect and counteract small-scale, dynamic cross-layer attack activities is now more important than ever.

The main issue with legacy security countermeasures is that they are usually tailored to address specific threats under specific network circumstances defined *a priori* [74]. On the other hand, the reality is that *malicious activities are usually extremely dynamic in nature and thus cannot be fully addressed beforehand*. As wireless attacks become ever more sophisticated, next-generation wireless networks will need to abandon generalized, one-size-fits-all, bolted-on security and optimization mechanisms, and rely on "smart", dynamic solutions able to harness the synergy between hardware and software reconfigurability to provide reliable, efficient and effective cross-platform and cross-layer security solutions. This aspect hinders significantly the integration and coordination of different wireless networking technologies to maximize network capacity, reliability, security, and coverage, and prevent the provision of a true networking-as-a-service vision. For this reason, software-defined radio techniques to simplify network control and to make it easier to introduce and deploy new applications and services, and machine learning to provide adaptability and fast-time reaction to adversarial action.

Recently, *machine learning techniques* have exhibited unprecedented success in classification problems in areas such as speech recognition [21], spam detection [18], computer vision [35], fraud detection [3], and computer networks [5], among others. One of the main reasons behind machine learning's popularity is that it provides a general framework to solve very complex classification problems where a model of the phenomenon being classified is too complex to derive or too dynamic to be summarized in mathematical terms [22, 46, 112]. Almost in parallel with machine learning's development, the development of algorithms and protocols based on *software-defined radios* [43, 114] has gained tremendous momentum in the networking research community over the last years [58]. A software-defined radio is a wireless communication system where components that have been typically implemented in hardware (e.g. mixers, filters, amplifiers, etc.) are implemented in software to ensure fast reconfigurability and adaptation to critical channel conditions (e.g., significant multipath, Doppler effect, or path loss). The main downside of pure software-based solutions is that they completely trade-off reconfigurability for efficiency. On the other hand, we have recently seen a tremendous rise of wireless platforms based on the *system-on-chip* (SoC) concept [59]. These SoC platforms allow the design and implementation of customized hardware on the field-programmable gate array (FPGA) portion of the platform to achieve better performance [63].

Although existing work has used machine learning and software-defined radios to design wireless security systems, these approaches have been used in an ad-hoc manner (i.e., to solve a specific wireless attack). On the contrary, future wireless networks will need to use context-aware, adaptive security measures able to sense the environment and swiftly respond to a range of dynamic, unpredictable, cross-layer attacks. We envision a radically different approach to the design of wireless security systems that can deploy various defense strategies, depending on the network's protection needs, and ability to tolerate and manage the specific technique's dynamic configuration.

In this chapter we address the lack of a unifying, systematic approach to cross-layer wireless security by proposing and discussing a *Learning-based Wireless Security* (LeWiS) framework. First, we provide background notions on the enabling technologies for LeWiS in Sect. 8.2. Then, we discuss a taxonomy of relevant existing wireless networks in Sect. 8.3. We then provide an overview of the LeWiS framework in Sect. 8.4, and delve deeper into the learning-based control module of LeWiS by discussing its detection (Sect. 8.5) and mitigation (Sect. 8.6) modules. We conclude the chapter in Sect. 8.7.

8.2 Background on Enabling Technologies for LeWiS

In this section, we provide a brief survey of the technologies that are at the basis of LeWiS, i.e., software-defined radios and networking (Sect. 8.2.1), system-on-chip technologies (Sect. 8.2.2), and machine learning (Sect. 8.2.3).

8.2.1 Software-Defined Radios and Networking

Software-defined radios are generally defined as devices where frequency band, air interface protocol and functionality can be upgraded with software updates instead of a complete hardware replacement. The peculiar characteristic of software-defined radios is that they are capable of being re-programmed or reconfigured to operate with different waveforms and protocols through dynamic loading of new waveforms and protocols. Furthermore, protocol characteristics at the data-link, network and transport are completely defined in software and may be changed dynamically at runtime depending on the system's needs. This has several advantages, including the ease of portability of the software to different hardware platforms, which in turn decreases development and testing costs as well as time-to-market of the radio technology. The software-defined radio technology uses modules that run on a generic hardware platform consisting of digital signal processing (DSP) processors as well as general purpose processors to implement the radio functions to transmit and receive signals. Usually, software-defined radio implementations are based on a combination of FPGA/DSP or FPGA-only solutions in alliance with software-defined physical and data-link layer for reaching the trade-off between parameter adaptation and performance. For an excellent and recent tutorial on the topic, the reader is referred to [99].

8.2.2 System-on-Chip (SoC) Technologies

System-on-chips (SoCs) [59] are embedded devices where a general-purpose processing system resides on the same integrated circuit with a field-programmable gate array (FPGA). SoCs are one of the leading technologies on the market for the implementation of digital systems combining software parts with hardware accelerators. The latest generations of these embedded devices include, among many other useful resources, powerful embedded hard processors supporting different operating systems, analog front-ends, specialized hardware blocks for high-performance computing or crypto-acceleration, and communication interfaces compatible with the most widely used network protocols.

The programmable logic implemented on the FPGA enables the efficient implementation of systems that perfectly fit the heterogeneous nature of wireless applications. This is because both hardware and software components can be configured according to the needs of different target applications; they are relatively low cost, low power, and compact, and their flexibility and possibility of code reuse (both hardware and software) allow the time to market to be reduced. Parallelism is another significant advantage of FPGAs. The distributed nature of the logic and interconnect resources in an FPGA fabric, together with the inherent concurrency of the hardware, allows several blocks operating in parallel (with either the same or different functionalities) to be implemented on a single chip.

8.2.3 Machine Learning

The pioneer of machine learning, Arthur Samuel, defined it as a "field of study that gives computers the ability to learn without being explicitly programmed"

[68]. Machine learning focuses on classification and prediction based on known properties, and usually consists of two phases: training and testing. Often, the following steps are performed: (1) identify class attributes (features) and classes from training data; (2) identify a subset of the attributes necessary for classification (i.e., dimensionality reduction); (3) learn the model using training data; and (4) use the trained model to classify the unknown data.

There are three main types of machine learning approaches: unsupervised, semi-supervised, and supervised. In unsupervised learning problems, the main task is to find patterns, structures, or knowledge in unlabeled data. When a portion of the data is labeled during acquisition of the data or by human experts, the problem is called semi-supervised learning. The addition of labeled data greatly helps to solve the problem. If the data are completely labeled, the problem is called supervised learning and generally the task is to find a function or model that explains the data. The approaches such as curve fitting or machine-learning methods are used to model the data to the underlying problem; the label is the problem variable. Recently, machine learning techniques have been used extensively in areas such as speech recognition [21], spam detection [18], computer vision [35], fraud detection [3], computer networks [5], and cyber intrusion detection [9], among others. The reason behind the popularity of machine learning is that it provides with a general framework to model and solve very complex problems. Furthermore, machine learning operates "on the fly" without requiring a model of the environment, the attacker's behavior, and with (almost) no human intervention. These characteristics make machine learning the ideal choice to detect stealthy, dynamic and unpredictable cross-layer wireless security threats.

8.3 Taxonomy of Existing Wireless Network Attacks

In this section, we provide an overview of the relevant wireless network attacks to date. Although wireless attacks are diverse in nature, it is possible to classify them into two well-distinct classes. i.e., *active* and *passive* attacks. The two classes have the following features:

- *Active attacks*: these attacks aim at partially (or completely) altering, corrupting or destructing ongoing communications. Typical examples of such an attack are *Denial of Service* (DoS) attacks such as *jamming* where the *jammer* exploits the broadcast nature of wireless communications to intentionally generate interfering signals that can potentially block all ongoing communications over one or more wireless channels. Other classical and highly disrupting attacks are *selective forwarding* and *reply attacks*. The former attack is generally aimed at mesh and relay-aided networks where a malicious node can selectively drop or forward a subset of the received packets, thus generating a decrease in the packet delivery ratio of the network. Instead, replay attacks aim at retransmitting multiple copies of the same packet. While wired networks can efficiently avoid such an attack by means of firewalls, wireless networks are particularly vulnerable to replay attacks, especially when cloned packets corresponds to user association, handshakes and ACK packets.[1]

- *Passive attacks*: in contrast with active attackers, passive attackers do not interact with the system actively but, instead, limit their actions to the monitoring of ongoing communications. A classical example of such an attack is *eavesdropping*, where a malicious node, the so-called *eavesdropper* aims at monitoring wireless channels to detect ongoing communications. When a communication is detected, the eavesdropper can either get access to the content of transmitted packets and compromise the confidentiality of data, or it can monitor RF transmissions to obtain statistical information on routes, identity of nodes and their transmission activity.

It is clear that passive attackers are very hard to be detected as they never disclose their presence or position. Accordingly, to protect the network from passive attacks, prevention mechanisms such as encryption [56, 100], steganography [24, 25, 55, 57, 79] and access control lists [70] are generally employed. The network is generally not able to measure the effectiveness of any prevention mechanism. Therefore, prevention mechanisms are generally proactive and static.

For the above reasons, in this book chapter we focus our attention on active attacks only. That is, we will delve into those cases where it is possible to detect the presence of attackers, and it is possible to monitor and estimate the impact of their attacks on the achievable performance of the network.

8.4 Learning-Based Wireless Security Framework (LeWiS): An Overview

Figure 8.1 shows a block diagram of LeWiS. The core of LeWiS lies in two different yet interconnected modules, i.e., the learning-based network stack (LNS) and the learning-based control module (LCM).

Learning-Based Network Stack (LNS) The main task of the LNS is to adapt transmissions protocols and radio frequency (RF) configurations to implement dynamic protocols based on control logic algorithms defined in the LCM. To this end, the LNS (1) swiftly implements the necessary configuration changes to address adversarial action; (2) increases network throughput as much as possible; and (3) eases the definition of protocols with cross-layer optimum behaviors. The LNS operates on the whole set of existing networking layers, and involves protocols from PHY to MAC to routing layers and to transport layer.

[1]For a more detailed discussion on active attacks and their impact on wireless communications we refer the interested readers to [4, 85, 107], where an exhaustive analysis of active attacks and their corresponding defence mechanisms is provided.

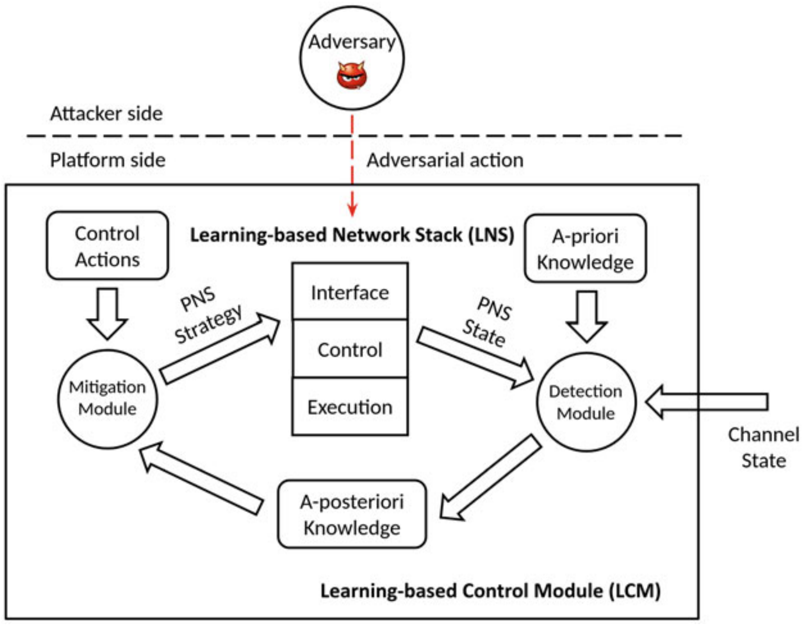

Fig. 8.1 Block scheme of the proposed Learning-based Wireless Security (LeWiS) framework

Although the logic of how network parameters are changed will be completely handled by the LCM, the LNS will be in charge of (1) actuating the commands received from the LCM; (2) coordinating the different network layers in a reliable and efficient way; and (3) storing and managing the current network state. Accordingly, the LNS is divided into three distinct but interacting "planes", each in charge of coordinating a different group of functionalities. We refer to these as the *execution* (where the protocols actually run), *control* (store network status information and receive parameters), and *interface* (provide a set of cross-platform APIs between the LCM and the LNS).

We now provide an overview of the LNS components and the interactions between them. We describe each component in a bottom-up fashion.

Execution Plane The execution plane (EP) handles the actual implementation of the networking algorithms. Specifically, it handles the set of data structures and algorithms, as well as the signal processing and RF front-end needed for wireless communication. As such, the EP is cross-layer and cross-domain by design, and resides on both the hardware (HW) and operating system (OS) portions of the wireless nodes.

The EP is structured as follows. The raw information bytes coming from the application (e.g., voice, video, data) are received, and after header formation at the transport (TSP), network (NET) and link (LNK) layers, the information packet is transformed into digital waveforms at the physical (PHY) layer and subsequently transmitted via the RF front-end. According to the commands provided by the control plane (CP), the EP chooses from a set of protocols and algorithms at every layer of the protocol stack. For example, the EP might choose different PHY schemes (e.g., OFDM, narrowband, spread-spectrum) or MAC schemes (e.g., CSMA, TDMA, ALOHA, etc.), according to what instructed by the CP.

The state of protocols at different layers are stored as variables in the CP. Therefore, any state variable of any protocol at any layer can be reconfigured on-the-fly through the CP. Protocol parameters (e.g., inter-frame spacing, modulation, carrier frequency, size of minimum contention window) can be "intrinsically" accessed by the CP as a result of a decision algorithm, or by the EP according to specific data acquisitions.

Control Plane The control plane (CP) is responsible for storing the network state composed by the set of parameters used at the different layers (e.g., modulation, transmit power, size of minimum CW, neighbouring list, and routing table). Furthermore, the CP is designed to handle the logic of data processing, which takes place in the EP. The CP decides the sequence of data processing functionalities that will be executed in the EP. The CP also controls switching between different protocols on-the-fly based on the decisions taken by the LeWiS mitigation module.

Interface Plane The interface plane provides a set of application program interface (API) between the LNS and the LCM. Specifically, the purpose of this layer is to (1) communicate to the LCM the state of the LNS held by the register and control plane; and (2) handle the reception of updated network parameters coming from the LCM. This plane can be implemented purely in software or also in hardware (e.g., FPGA), depending on the platform's needs.

Learning-Based Control Module (LCM) The purpose of the LCM is to implement the algorithms that will ultimately provide the LNS with appropriate network parameters at every layer of the protocol stack. The choice of network parameters will depend on (1) the current environmental conditions; and (2) the current network state, provided by the LNS. This information is used by the LCM to take appropriate action in response to various phenomena, ranging from a change in network objectives (e.g., more throughput is needed at the cost of energy consumption) to a detection of an ongoing network attack.

The LCM provides *decisions* based on user-defined machine learning algorithms. Such "decisions" may be different in nature; they include, for example, modifying a parameter in a protocol (e.g., transmit power), switching to a different transmission scheme at the PHY layer (e.g., from OFDM to CDMA), and complex cross-layer decision making such as joint routing and spectrum allocation, among others. The output of the LCM is not applied to the LNS directly, but specific APIs are used in the Interface plane to establish communication between the LCM with the LNS. Therefore, results of specific machine learning algorithms can be accessed by the Execution and Control planes of the LNS and adopted in their logic. Thus, *the LCM separates the decision plane from execution of the protocol stack, which enables the* *definition and reconfiguration of the decision logic on-the-fly without influencing the on-going protocol execution logic*. Most importantly, this separation provides the capability to (1) apply locally control decisions taken at other devices; and (2) create decisions to be applied at other network devices.

From a logical standpoint, the LCM is composed by a *detection* and a *mitigation* module. The detection module records a series of observed events and use them to update the a priori distribution of the parameters of the underlying network state. The resultant a posteriori distribution is fed to the mitigation module, which in turn decides the strategy to address the ongoing attack. Another task of the LCM is to sense the environment and acquire as much information as possible regarding the outside world. This includes carrier sensing, detection of number of nodes, and so on. This information will be fed to the machine learning algorithms, and if necessary used for training the models.

The rest of this section will be devoted to describe in details the LeWiS detection and mitigation modules. By doing so, we will also discuss the relevant state of the art on the topic of detection and mitigation of wireless attacks.

8.5 LeWiS Detection Module

In this section, we discuss the detection module of LeWiS. First, we introduce the general structure of the LeWiS detection module. Then, the challenges of detection are discussed. Next, we review different machine learning techniques that can be used in the LeWiS detection module. Finally, we discuss different detection approaches.

The objective of the detection module is to use machine learning techniques to construct reliable hypotheses on the existence of possible attacks in a network, based on observed evidences and *a-priori* knowledge on the attacks. The detection results are represented in the form of *a-posteriori* knowledge on the attacks and supplied to the mitigation module for defense strategy making.

The detection module is made up by three components, an event monitor, a learning engine, and a learning controller, as is shown in Fig. 8.2. We discuss each component below.

- *Event Monitor.* The event monitor observes and records events occurred on multiple layers that are useful for the detection of the underlying network state. Although an attack can be identified using the network state changes made by it, states are not always directly observable. On the other hand, there are always observable events whose occurrences are a function of the underlying network state. These event may include a successful (or failed) reception of a bit (or packet), the result of a channel contention, and the end-to-end delay of a message, among others. These events may reveal the underlying network state often in a *probabilistic* way.
- *Learning Engine.* The learning engine computes the probability that (1) a certain attack is taking place (supervised learning techniques), or (2) the belief that the network is running in a "normal" state (unsupervised learning techniques).

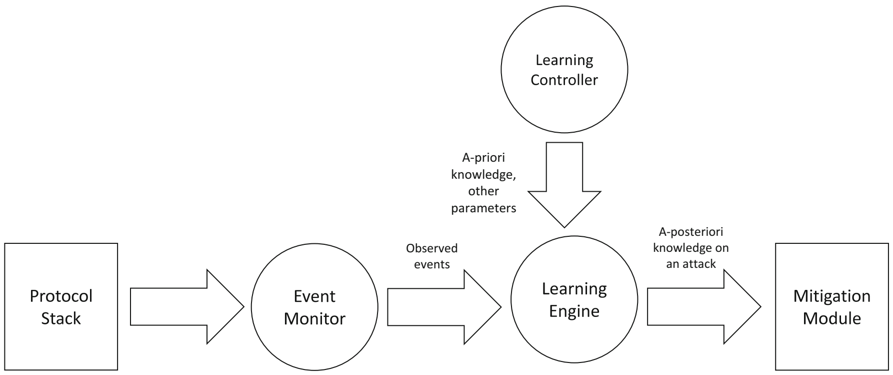

Fig. 8.2 Components of the detection module

Specifically, it takes the events observed by the event monitor as the input, generates a "data" point based on the events, which represents the network state. Then, learning techniques such as K nearest neighbor (KNN), support vector machine (SVM), or clustering are used upon the data point. In most cases, it also takes *a-priori* knowledge in some form as an input. The *a-priori* knowledge represents the knowledge on the expected behavior of an attack, and can be derived by theoretical analysis, or, more importantly, training.

- *Learning Controller*. The learning controller is mainly in charge of the parameter settings for the learning engine. Depending on the application and implementation of a network, it may be prone to different attacks. Meanwhile, depending on the threat of an attack, it may not always be worthy defending it, considering the overhead. This is especially important when the number of possible attacks are high. Therefore, the learning controller should decide which attacks to detect, and consequently, which layers and events to monitor. The learning controller is also in charge of feeding the learning engine with the proper *a-priori* knowledge.

8.5.1 Challenges in Cross-Layer Attack Detection

There are different ways to launch cross-layer attacks. "Sub-attacks" on different layers with similar consequences may be used *in parallel* for a better result. For example, observing that denial-of-service can be achieved using multiple attacks on different layers, it is argued in [81] that a cross-layer DoS attacks can be launched by jointly utilizing these sub-attacks. On the other hand, it is also possible for a sophisticated attacker to launch a more advanced cross-layer attack by exploiting the cross-layer interactions in the underlying protocols of the network (in the following contents, we will deliberately refer to this type of attacks as "advanced" cross-layer attacks when distinguishing is needed). Compared to traditional single-layer attacks, these advanced cross-layer attacks have several unique challenges:

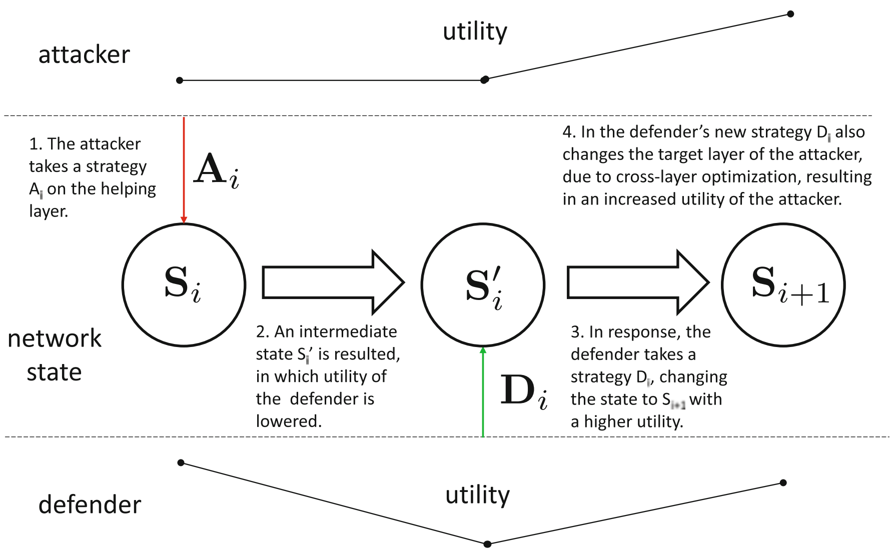
attacker
utility
1. The attacker takes a strategy A_i on the helping layer.
$\mathbf{A}_i$
4. In the defender's new strategy D_i also changes the target layer of the attacker, due to cross-layer optimization, resulting in an increased utility of the attacker.
$\mathbf{S}_i$
$\mathbf{S}'_i$
$\mathbf{S}_{i+1}$
network state
2. An intermediate state S_i′ is resulted, in which utility of the defender is lowered.
$\mathbf{D}_i$
3. In response, the defender takes a strategy D_i, changing the state to S_i+1 with a higher utility.
defender
utility

Fig. 8.3 The mechanism of cross-layer attacks

- *Stealthiness.* The major issue in cross-layer attack detection is overcoming their stealthiness. To explain this point, let us consider the example shown in Fig. 8.3. We define as *network state* a state of network variables, which may include signal-to-interference-plus-noise ratio (SINR) of a link, the channel access probability of a node, and the quality of a route, among others. Let us suppose the network is in a state $\mathbf{S}_i$, and that the adversary aims at changing a certain *target* layer so that the network enters a new state where the defender is penalized. To this end, instead of attacking the target layer directly as in a traditional single-layer attack, in this case the adversary chooses another layer, i.e., the *helping* layer, and attacks it with a strategy $\mathbf{A}_i$. This causes the defender to switch to an intermediate state $\mathbf{S}'_i$ where the utility of the defender is lowered. As a response, the defender chooses a strategy $\mathbf{D}_i$ to switch to a more favorable state $\mathbf{S}_{i+1}$.

  Since the defender jointly optimizes its strategy on multiple layers, strategy $\mathbf{D}_i$ changes the target layer of the adversary as well, as long as the attack strategy $\mathbf{A}_i$ is carefully chosen. In other words, with a fine-tuned attack strategy, a cross-layer attack can create a state in which the defender's responsive strategy also benefits the adversary. Therefore, cross-layer attacks have the potential to be exceptionally *stealthy*, and how to detect the often small-scale attack activities becomes the most important challenge.
- *Heterogeneity.* Cross-layer attacks involve multiple "sub-attacks" on different layers. Therefore, different cross-layer attacks may behave similarly on a specific layer, especially on the helping layer. This suggests high heterogeneity in the form of attacks even if similar attack patterns have been observed.

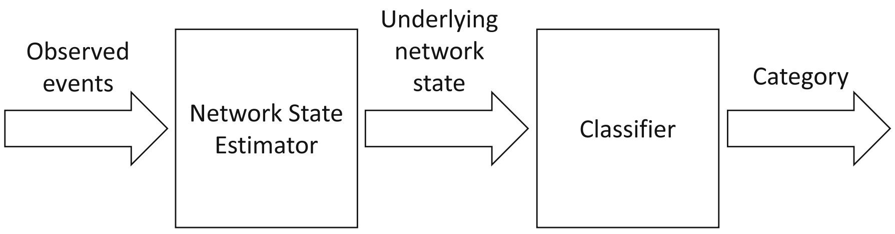

Fig. 8.4 Learning engine

The challenge of stealthiness suggests that the network state that distinguishes an attack is difficult to estimate; and the challenge of heterogeneity suggests that it is difficult to distinguish different attacks. In the learning engine, these challenges are addressed by two components, a network state estimator and a classifier, as shown in Fig. 8.4. The network state estimator estimates the underlying network state that may be ambiguous to a "normal" state due to small-scale attack activities, and generates a feature vector representing the current state. The classifier is trained to classify such a feature vector to a specific attack (misuse detection) or abnormal (anomaly detection).

8.5.2 Learning Techniques for Attack Detection

Various machine learning techniques can be applied to accomplish the tasks of both network state estimation and classification.

8.5.2.1 Network State Estimation Techniques

Network state estimator is tasked to estimate the underlying network state, which is usually not directly observable and may only be changed by the attacker slightly. It falls to the topic of parameter estimation in machine learning, and maximum likelihood (ML) estimation and Bayesian estimation are two widely-used methods [27].

ML Estimation ML estimation aims at finding the candidate hypothesis that maximizing the likelihood for the observed events to happen. Specifically, suppose the probability (likelihood) for an event $\mathbf{E}_k = \mathbf{e}_k$ to happen for the underlying network state $\mathbf{S} = \mathbf{s}$ is

$$\mathbb{P}\{\mathbf{E}_k = \mathbf{e}_k | \mathbf{S} = \mathbf{s}\} = f(\mathbf{e}_k, \mathbf{s}), \tag{8.1}$$

where the function $f(\cdot)$ is available by either theoretical analysis or training, then the ML estimator generates the hypothesis $\mathbf{S} = \mathbf{s}$ if

$$\mathbf{s} = \arg\max_{\mathbf{s} \in \mathscr{S}} f(\mathbf{e}_k, \mathbf{s}), \tag{8.2}$$

where $\mathscr{S}$ is the set of all possible values for the network state $\mathbf{S}$.

ML estimation treats the network state as a fixed value and finds the value best fits the evidences. It does not take into account the a-priori knowledge on the network state, which is usually available. Therefore, in such cases, it is not as good as Bayesian estimation, which is based on a-posteriori instead of likelihood. However, it is relatively less complex, especially when dealing with binary hypotheses. For example, log likelihood ratio test (LLR) is used for jamming detection [77].

Bayesian Estimation Bayesian estimation, on the other hand, treats the network state $\mathbf{S}$ as a random variable and constructs *a-posteriori* distribution. It takes into account the a-priori knowledge on the network state. Besides, as a distribution on the set of all possible values instead of the likelihood for one single value, Bayesian estimation provides more information on the network state than ML estimation. Therefore, for cross-layer attacks creating a network state $\mathbf{S}'$ that is only slightly different from the "normal" state $\mathbf{S}$, Bayesian estimation is generally a better choice than ML estimation.

Specifically, with a sequence of independent events $\{\mathbf{e}_k\}_{k=1,\ldots,K}$ have happened, it follows that

$$\mathbb{P}\{\{\mathbf{E}_k\}_{k=1,\ldots,K} = \{\mathbf{e}_k\}_{k=1,\ldots,K} | \mathbf{S} = \mathbf{s}\} = \prod_{k=1}^{K} f(\mathbf{e}_k, \mathbf{s}), \tag{8.3}$$

which leads to the following *a-posteriori* distribution

$$\begin{aligned} &\mathbb{P}\{\mathbf{S} = \mathbf{s} | \{\mathbf{E}_k\} = \{\mathbf{e}_k\}\} \\ &= \frac{\mathbb{P}\{\{\mathbf{E}_k\} = \{\mathbf{e}_k\} | \mathbf{S} = \mathbf{s}\}}{\int_{\mathbf{s}} \mathbb{P}\{\{\mathbf{E}_k\} = \{\mathbf{e}_k\} | \mathbf{S} = \mathbf{s}\} \cdot \mathbb{P}\{\mathbf{S} = \mathbf{s}\}\, \mathrm{d}\mathbf{s}} \cdot \mathbb{P}\{\mathbf{S} = \mathbf{s}\}, \end{aligned} \tag{8.4}$$

where $\mathbb{P}\{\mathbf{S} = \mathbf{s}\}$ is some *a-priori* distribution of $\mathbf{S}$, which represents the "normal" network state (i.e., without attacks). Note that such quantity is often available—for example, the distribution of SINR on a link can be derived from the fading model, the channel access probability for any node in a network running CSMA/CA is approximately the same, and so on. If accurate knowledge is not available, it is often possible to know some information on it, such as the functional form, and the range of its values [27].

The *a-posteriori* distribution in Eq. (8.4) reveals the possible underlying network state, so the defender is now aware of the attack *activities* of the attacker. The resultant distribution is fed to a classifier to classify a new data point. For example, with the signature of a suspected attack, we may have the classifier in the following general form:

$$C(\mathbb{P}\{\mathbf{S}=\mathbf{s}|\{\mathbf{E}_k\}=\{\mathbf{e}_k\}\}, \mathbf{F}) \underset{\text{Not attacked}}{\overset{\text{Attacked}}{\lesseqgtr}} C_{\text{th}}, \tag{8.5}$$

where **F** represents the signature of the suspected attack, and C_{th} is a threshold that is set based on an *a-priori* training of the system. The detailed form of the classifier depends on the targeting attack, and the classification techniques. We will discuss the techniques in the following subsection.

8.5.2.2 Classification Techniques

There is a rich body of classification (or clustering for unsupervised learning) techniques used in network intrusion detection systems (NIDS). [45] gives an experimental comparison on such techniques applied to the KDD dataset [41]. KDD dataset is a famous benchmark dataset for network intrusion detection. It is packet-based, involving packets generated on network layer and above. The attacks are mainly single-layer attacks. However, the machine learning techniques still have the potential to be applied for the classification of cross-layer attacks. We have listed in Table. 8.1 the most widely-used techniques and the attacks they have been applied to.

K Nearest Neighbor (KNN) K Nearest Neighbor (KNN) is an instance-based learning technique, which classifies a data point based on the majority of the classes of its K nearest neighbors (usually according to the Euclidean distance in the feature space).

Table 8.1 Learning techniques used in attack detection

	Techniques	Attacks
Supervised	K nearest neighbors (KNN)	KDD attacks[a] [48], flooding [50]
	Decision tree	KDD attacks [73, 76]
	Support vector machine (SVM)	Sinking [37]
	Artificial neural network	KDD attacks [6, 53, 88], DDoS [67]
	Ensemble learning	KDD attacks [1, 32]
Unsupervised	Clustering	DDoS [47]
	K means	KDD attacks [52]
Reinforcement	Q-learning	Not specified [34, 72]

[a]We refer to KDD attacks the set of attacks in KDD dataset, including DoS, user to root (U2R), remote to local (R2L), and PROBE attacks

Formally, denoting the set of the K nearest data points as $\mathscr{K}$, then the probability that a new data point $\mathbf{x}$ belongs to a class y is computed as

$$\mathbb{P}\{y|\mathbf{x}\} = \frac{1}{K} \sum_{x_i \in \mathscr{K}} I(y_i = y), \tag{8.6}$$

where y_i is the label of data point x_i, and $I(y_i = y)$ is an indicator function for the condition $y_i = y$. The classifier is then assigns class y_* according to

$$y^* = \arg\max_{y \in \mathscr{Y}} \mathbf{P}\{y|\mathbf{x}\}, \tag{8.7}$$

with $\mathscr{Y}$ denoting the set of all classes.

KNN is one of the least complex machine learning techniques and thus has the potential to be applied in wireless networks with low cost devices. It has been evaluated on wireless sensor networks in [50] for flooding attack and used on the KDD dataset in [48].

Decision Tree Decision tree is a technique to organize multiple rules in a tree-like model. Each node in the tree except the leaves corresponds to a test over some attributes (or features) of a data point. The decision process is directed to different children based on the test result, until a leaf node is reached, which represents a decision. Decision tree techniques such as C4.5, random tree, and random forest have been applied to intrusion detection on the KDD dataset in [73, 76].

Support Vector Machine (SVM) Support Vector Machine is based on the assumption that data points for different classes are *separable* when represented in a high dimension feature space. The data points are first mapped to a high-dimensional feature space. Then, a linear decision function is constructed in the mapped feature space, resulting in hyperplanes separating two classes of data points.

Specifically, suppose the mapped data points is represented as a vector $\mathbf{x}$, then the classifier is in the form of

$$\mathbf{w}^T\mathbf{x} - b \geq 1 \tag{8.8}$$

for one class, and

$$\mathbf{w}^T\mathbf{x} - b \leq -1 \tag{8.9}$$

for the other. The weight $\mathbf{w}$ should be computed to maximize $\frac{2}{\|\mathbf{w}\|}$, i.e., the margin between the two classes.

In [37], SVM is used for the detection of sinking attack in wireless ad-hoc networks.

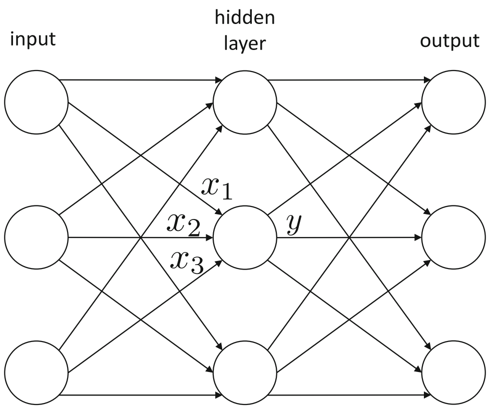

Fig. 8.5 Artificial neural network

Artificial Neural Network Artificial neural networks are inspired by biological neural networks. It is a network of artificial neurons (nodes), as shown in Fig. 8.5. "signals" are transmitted through connections between the neurons. At each neuron, the "signal" is processed using a (usually non-linear) function of the weighted sum of all inputs. Denoting the input as **x** and the weight as **w**, the output y is

$$y = f(\mathbf{w}^T\mathbf{x}), \tag{8.10}$$

where the function $f(\cdot)$ may take different forms. There are typically multiple layers of neurons, with the first layer as the input and the last layer the output (the classification result). The weights in a neural network are trained to produce the most favorable outputs. It has found applications in detection of various attacks in [6, 53, 67, 90].

Ensemble Learning Ensemble learning techniques utilize the hypotheses generated by multiple weak learners to construct a strong one that outperforms each individual weak learner. There are multiple ways to ensemble weak learners, such as bagging and boosting [27]. In detection of cross-layer attacks, due to the heterogeneity in sub-attacks, the best learning techniques in detecting each of them may vary. Therefore, a good ensemble method may improve the performance of the detection. An AdaBoost-based learning with decision tree as weak learners is evaluated on KDD dataset in [32]. In [1] a simple ensemble method of weighted majority is used.

Clustering Clustering is an unsupervised learning technique that allows the unlabeled data points form different groups (clusters) automatically based on their features. Different methods can be used for this purpose. In [47] a hierarchical clustering method is used to detect DDoS attacks.

K Means K means is a classic clustering method. It first randomly creates K clusters and then iteratively updates the center of each cluster with new data points until convergence. New data points can be assigned to a cluster according to the distance.

For standard K means, suppose the mean (center) of cluster k at step i is $\mathbf{m}_k^i$, then a data point $\mathbf{x}_j$ is assigned to cluster

$$k^* = \underset{1\leq k\leq K}{\arg\max} \|\mathbf{x}_j - \mathbf{m}_k^i\|. \tag{8.11}$$

With all data points clustered, the new means are computed as

$$\mathbf{m}_k^{i+1} = \frac{1}{\|C_k^i\|} \sum_{j\in C_k^i} \mathbf{x}_j, \tag{8.12}$$

with C_k^i denoting the set of data points belonging to cluster k at step i. The update stops when there is only negligible changes in the means.

In [52] K means is used with KNN for intrusion detection, where K means is used to form clusters and KNN is used to assign new data points to the clusters.

Q-Learning Instead of aiming at classifying (or clustering) data points, reinforcement learning aims at finding the optimal policy to maximize the reward in a dynamic system with multiple states. Q-learning is a popular reinforcement learning technique, which can be implemented using dynamic programming. As a decision making process, it fits more for the task of attack mitigation (introduced with more details in Sect. 8.6), but there are still several works in applying Q-learning to attack detection [34, 72]. The common idea is that the detection and attack form a game, in which the defender and attacker may adopt different actions (to detect or not to detect for the defender; to attack or not to attack for the attacker) and get different rewards. Q-learning can be used to find the optimal policy for each side in this game-theoretical framework. Note this is a high-level model, and the actions for both sides are abstract, without detailed detection (or attack) methods involved, therefore, Q-learning (and reinforcement learning techniques in general) needs to be used with other detection methods dedicated for the attacks for a good performance.

8.5.2.3 Detection Approaches: Misuse vs Anomaly

We have reviewed different techniques that can be used in attack detection. An equally important question in detection module design is the detection approach. There are two approaches in intrusion detection, misuse detection and anomaly detection [42]. The former aims at identifying a specific attack based on its signature, while the latter aims ate finding out outliers to “normality”.

The framework established in [106] has adopted the misuse detection approach. It is argued that the learning engine should be tailored for each target attack individually. However, there are also scenarios where it is difficult to adopt this approach. For example, in networks with low cost devices and low security requirement, it is neither feasible nor necessary to monitor multiple advanced cross-layer attacks; in scenarios lacking knowledge on the potential attacks, it is also difficult to train a model for the unknown attack. In such cases, the anomaly detection approach seems more promising.

The common idea of anomaly detection is to monitor multi-layers and cluster the data points into different groups. Only one of the groups is considered normal, and for data points in other groups, the mitigation module will be called to "revert" the state back to normal. This step may contribute to the launching of an advanced cross-layer attack, as shown in step 3 in Fig. 8.3. Therefore, there is a possibility that anomaly detection fail against advanced cross-layer attacks.

Reference [87] is a typical example of this idea. It is assumed that there are intrinsic and observable distinction between normal and abnormal behaviors. The authors propose to select "features" that best distinguish normal and abnormal behaviors on multiple layers, and use classifiers such as decision tree, Bayesian Network, and SVM to classify if a data point is normal or not.

It is generally believed that anomaly detection is able to detect not only already-known attacks but also unknown attacks. However, in [75] it is argued that this may not hold in practice, due to the unsuitability to use machine learning in outlier detection, the high cost of errors, the semantic gap between the anomaly detection results and the insight on defense, among others. Therefore, anomaly detection may only enjoy the advantage of simplicity and low cost compared to misuse detection.

8.6 LeWiS Mitigation Module

The primary goal of the mitigation module is to efficiently counteract ongoing attacks by selecting and combining one or more defence strategies among a set of feasible defence mechanisms. In order to identify effective defence strategies, the mitigation module needs to address the following core challenges: (1) attacks can be launched by a potentially large number of adversaries; (2) the adversaries might be heterogeneous and be able to attack the network from multiple locations of the network; and (3) their behavior (i.e., their attack strategy, position, and so on) and number might vary over time.

The above challenges are peculiarly hard to be tackled. Indeed, it has been shown that attackers can easily hide their location by exploiting the broadcast nature of RF communications [26, 49], and can also maximize the undetectability and impact of their attacks by using simple but effective attacks. As an example,*pilot jamming* [15, 71] and *pilot nulling* [15] can be used to attack pilot tones and partially or completely corrupt synchronization and equalization operations at the receiver side. Another example is *reactive jamming* [25, 93], where the jammer uses energy-detector systems to first detect ongoing communications, and then transmit interfering signals aimed at disrupting RF transmissions.

This discussion shows that it is highly difficult (if not impossible) to fully characterize a network of attackers and deterministically derive the best defense strategy. Accordingly, not only is LeWiS tasked to learn how to defend the network from a particular attack, but it has also to adapt to the heterogeneous and ever-changing environment and attacks. Therefore, our framework LeWiS relies on a learning-based mitigation module where *a-priori* knowledge, real-time observations and supervised control actions are jointly leveraged to continuously adapt to network attacks and to derive appropriate defence mechanisms for any given network state and topology.

To provide the network with proper defense mechanisms, LeWiS continuously relies on the three components described below, and keeps tracking present and past network states. Also, the detection module notifies the mitigation module with relevant information on the nature of ongoing attacks. Accordingly, the mitigation module is capable of automatically computing the most effective defense strategy to mitigate, and possibly avoid, further attacks.

- *A-priori knowledge.* It is a set of static pre-loaded defence strategies for a given subset of attacks. It is used by the mitigation module to counteract ongoing attacks in the bootstrapping stage of LeWiS, i.e., when no information regarding the presence of attackers is available. Thus, as soon as an attack is detected, the mitigation module leverages the a-priori knowledge to identify one or more suitable defence strategies;
- *A-posteriori knowledge*: This set of strategies is continuously and autonomously built, updated and enhanced by LeWiS by evaluating the effectiveness of the different defence strategies employed in the past. Also, it is used by LeWiS to update the state of the network and to keep track of ongoing attacks. Indeed, it must be assumed that the environment and the attacker might change over time. As an example, a jammer can move from a location to another, of adapt its attack strategy to target a specific layer of the protocol stack. Therefore, the system must always monitor the environment to update the state and gather knowledge;
- *Supervised Control Actions*: The efficiency of the mitigation module can be improved by introducing human-generated inputs that either add new defence strategies to the system, or can steer the learning algorithm toward one solution rather than another. Specifically, it is possible for the network operator to interact with the decision mechanism by (1) specifying control operatives and performance metrics (e.g., throughput maximization, delay and energy minimization, minimum QoS guarantees, maximum transmission power, etc . . .); and (2) introducing new defence strategies when a novel attack is discovered and thus cannot yet be detected by LeWiS. These inputs are then used by LeWiS to identify the best defence strategy that optimizes a given control operative while satisfying a given set of constraints or requirements.

8.6.1 Traditional Mitigation Strategies

Given the cross-layer nature of attacks, LeWiS combines different defence strategies involving multiple layers of the protocol stack. As an example, to overcome reactive jamming attacks (i.e., the jammer is triggered when the received signal's power is above a given triggering threshold), defence strategies might involve the use of defense strategies at the crossroads of different layer. For example, power control (i.e., a physical layer strategy) can be used to reduce the strength of the signal received by the jammer, and channel access control (i.e., a link layer strategy) can be used to avoid the subset of channels monitored by the jammer [26].

To effectively address the variety of attacks, the mitigation module relies on a modular approach where atomic and single-layer defence strategies are selected and then properly combined to address different attacks. For the sake of clarity, in Table 8.2 we provide a brief summary of possible defences at each layer of the protocol stack.[2]

Table 8.2 Defence mechanisms and their application in wireless networks

	Physical layer	Data link layer	Network layer	Transport layer	Application layer
Power control	[26, 30, 89]				
Modulation	[98]				
Coding	[62]				
Beamforming	[29, 102]				
Friendly jamming	[33]				
Artificial noise	[62]				
Medium access control (MAC)		[26, 40, 97]			
Relaying and routing		[44, 62]	[39, 69]		
Authentication	[83, 101]	[40]	[7, 39]		[19]
Load balancing	[14]		[2]	[14]	
Transport layer security (TLS)				[66]	
Secure socket layers (SSL)				[66]	
Firewall					[94]
Encryption	[20, 54, 60, 61]	[17, 36]	[17, 54]	[17]	[94]

[2]The provided summary is not intended to be exhaustive, and a more detailed taxonomy of defense strategies can be found in [13, 96]. Furthermore, the same defence strategy (e.g., authentication, coding) can be implemented in different ways, and with possibly distinct outcomes, at multiple layers of the protocol stack.

The majority of prior work in network security has put its efforts on the design and implementation of ad hoc defence solutions for a small subset of network attacks that are targeted at the lower layers of the protocol stack [30, 39, 62]. On one hand, this deterministic approach produces highly reliable and robust solutions to a subset of well-defined attacks. On the other hand, it is clear that this paradigm cannot be equally productive in the case of cross-layer attacks where (1) the presence, position and attack strategy of the attackers is unavailable; and (2) different attacks can be combined together to attack multiple layers of the protocol stack.

The technological advancement and the availability of relatively cheap but powerful reprogrammable devices, such as SDRs and FPGAs, has enabled and facilitated the development and spread of complex and cross-layer attacks [12, 31, 106]. For this reason, ad hoc defence mechanisms addressing only a small number of attacks are expected to achieve poor performance—in other words, a single "one-fits-all" defense strategy is unlikely to be found. Instead, research efforts should be funneled towards the definition of a system capable of adaptively counteracting ongoing attacks through the generation of cross-layer strategies obtained by properly combining single-layer defense mechanisms. This latter research challenge will be the focus of the next section.

8.6.2 Learning-Aided Mitigation Techniques

To provide a reliable defence framework, information with respect to the attackers and their attack strategies is indeed required. Although many works in the literature assume that such an information is available to the defender, such requirement cannot be always satisfied in a significant number of wireless network scenarios where only incomplete and possibly erroneous information is available. To overcome the lack of information, statistical information and learning techniques can be profitably used to learn the environment, test different defense strategies and subsequently identify the most effective ones.

Since the system aims at providing a secure and tamper-proof communication, it is possible to model the mitigation module of LeWiS as an *agent* whose objective is to (1) select the most appropriate set of the defense strategies, such that (2) a given the security level and the performance of the network are simultaneously maximized over a large time horizon. In this context, the mitigation problem can be modeled as a *Markov Decision Process* (MDP) [92]. Specifically, the MDP corresponding to the mitigation module is shown in Fig. 8.6 and relies the four following fundamental elements:

- *Action Set*: it is represented by the set $\mathscr{A}$ of all feasible defense strategies. Each defence strategy $a \in \mathscr{A}$ is stored as single-layer atomic action that can be successively combined with other actions, belonging to the same or to different layers, to counteract cross-layer attacks;

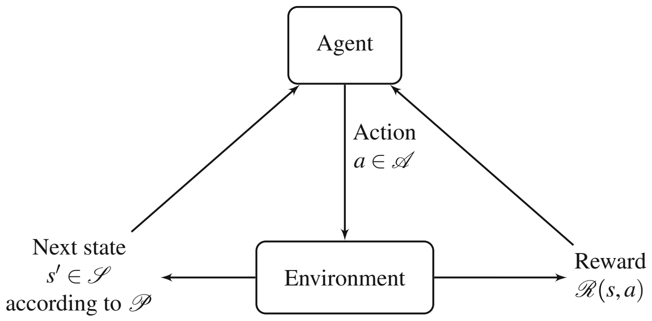

Fig. 8.6 Interactions in an MDP

- *State Set*: it represents the set $\mathscr{S}$ of possible states of the network, i.e., the network. As an example, each state $s \in \mathscr{S}$ can be used to represent network performance information such as bit, symbol and packet error rates, spectral efficiency and noise level. Furthermore, the state of the network also include information with respect to the type of attack, number of attackers and their position;
- *Reward Function*: this is a function $\mathscr{R} : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ that is used to represent the performance of the system when the network is in a given state and a given set of defence strategies are deployed. Typical examples of reward functions are throughput, data-rate, energy consumption and latency. The aforementioned metrics are used by the MM of LeWiS to evaluate the effectiveness of current defense strategies.
- *Transition Function*: a function used to regulate and describe the transition between a state to another. This function can be expressed as $\mathscr{P} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to [0, 1]$ and represents the probability $\Pr\{s'|(a, s)\}$ that the network enters state s' when the action a is taken by the decision maker when the network is in state s.

ML technologies, have been profitably used in the literature to address a variety of cross-layer optimization problems ranging from rate maximization [16, 108], channel estimation [28, 86] and resource allocation [5, 80]. Preliminary works on the application of ML algorithms to address security issues in wireless networks are already available in the literature [5, 9, 65, 103], however they are generally targeted at mitigating only a limited subset of attacks, thus not providing effective and comprehensive solutions to counteract heterogeneous cross-layer attacks in wireless networks.

Given the potential of ML techniques that make it possible to learn from the environment and adapt defense strategies accordingly, we envision a cross-layer and comprehensive defense system where any attack can be mitigated, or possibly avoided, by wisely tuning the learning parameters of the system.

ML technologies can be generalized in two main classes, namely *Dynamic Programming* (DP) and *Reinforcement Learning* (RL), whose features are as follows:

- **Dynamic Programming**: this is a model-based approach where the impact of a given action on the transition from a state to another and the corresponding obtained reward is a known a-priori [11]. That is, DP requires knowledge of the reward and transition functions $\mathscr{R}$ and $\mathscr{P}$, respectively.
- **Reinforcement Learning**: this is a model-free approach where the learning process builds its own knowledge by means of observations and exploration of previous actions and rewards. That is, RL does not require any a-priori knowledge of $\mathscr{R}$ and $\mathscr{P}$ as the information related to the transition from a state to another and the achieved reward is discovered by the learning process itself [11].

The two above approaches have been widely and successfully used in the literature to derive optimal control policies for many networking problems. However, the above analysis clearly shows that DP require some form of knowledge with respect to the underlying MDP, a condition that might not be guaranteed in many network scenarios. On the contrary, RL iteratively constructs and gathers knowledge by exploring the environment and the action space, thus making it a promising technology to design a reliable and efficient MM when accurate information on the attacker is not available.

In the following, we present few examples that show how both DP and RL can be effectively used to counteract attacks in wireless networks.

8.6.2.1 DP and Its Application to Attack Mitigation

Dynamic programming is a well-established learning approach [8] that makes it possible to generally derive optimal solutions to a variety of NP-hard problems [95].

To compute efficient mitigation strategies, DP relies on the following well-known *Bellman's Equation*:

$$J_t(s_t) = \max_{a_t \in \mathscr{A}} \underbrace{\mathscr{R}_t(s_t, a_t)}_{\text{Single}-\text{slot Reward}} + \underbrace{\mathbb{E}_{\mathscr{P}}\{J_t(s_{t+1})|(s_t, a_t)\}}_{\text{Cumulative Expected Reward}} \tag{8.13}$$

As already outlined in the previous section, Eq. (8.13) shows that DP strongly depends on the state transition probabilities $\mathscr{P}$. Intuitively, the function $J_t(s_t)$ is iteratively maximized at each iteration by considering the best action a_t in the set $\mathscr{A}$ such that the single-slot reward $\mathscr{R}_t(s_t, a_t)$ and the cumulative expected future reward of network is maximized.

Apart from being dependent from the transition function $\mathscr{P}$, another drawback of DP lies in the so-called *curse of dimensionality* [8]. That is, to compute optimal defense strategies, DP generally needs to test a large number of actions (e.g., channel assignment, power allocation, encryption, beamforming, etc...), which eventually results in high computational complexity when the number of actions is large and heterogeneous. As an example, it has been shown [26] that even a relatively simple cross-layer defense strategy that merges physical and data-link layer defense mechanisms, e.g., power control and channel allocation, generally results in NP-hard solutions. Specifically, to counteract reactive jamming attacks, DP requires exponential time to compute an effective defense strategy even if the continuous transmission power levels are quantized and substituted with their corresponding discrete variables. To overcome such an issue, a promising approach is to leverage on the learning engine of DP and exclude inefficient defense strategies from the action set [26]. As an example, if a given defence strategy is known to be either nonfunctional or inefficient, e.g., the network is aware that all transmission power levels above a given triggering threshold always activate the jammer, those actions can be removed from the action set. Such an approach not only reduces the complexity of the overall learning algorithm, but it also avoids poor performance due the deployment of ineffective defense strategies.

8.6.2.2 RL and Its Application to Attack Mitigation

In the context of RL many algorithms have been proposed to derive optimal and sub-optimal policies for a variety of optimization problems [10, 38, 78]. Those algorithms have different properties and provide different performance levels in different network scenarios. For the sake of simplicity, in this chapter we will focus our attention on two well-known and well-established RL algorithms, namely *Q-Learning* and *State–action–reward–state–action*.

- **Q-Learning** : this learning approach relies on the well-known *Q-function*. Let a_t and s_t be the action taken by the agent and the state of the system at time t, respectively. The Q-function is defined as follows:

$$Q(s_t, a_t) = (1 - \alpha_t(s_t, a_t))Q(s_t, a_t) + \alpha_t(s_t, a_t)\left[\mathscr{R}_t + \gamma \max_{a^* \in \mathscr{A}} Q(s_{t+1}, a^*)\right] \tag{8.14}$$

  where $\alpha_t(s_r, a_t) \in [0, 1)$ is the *learning rate* of the algorithm associated to the 2-tuple (a_t, s_t), and $\gamma \in (0, 1)$ is the *discount parameter* and s_{t+1} is the observed state of the network when action a_t was taken. Intuitively, (8.14) iteratively aims at maximizing the total discounted reward of the network, which in our case consists in the maximization of network performance through the mitigation of network attacks. It is worth mentioning that there are no particular restrictions on the action selection mechanism to be used in the learning process. Well-established and widely used action selection mechanisms are random selection, ε-greedy and softmax algorithms [78]. However, other approaches are available, and they can be found in [82]. It has been shown [91] that if the reward function is bounded and $\sum_{t=1}^{+\infty} \alpha_t^2(s_t, a_t) < \sum_{t=1}^{+\infty} \alpha_t(s_r, a_t) = +\infty$ for all $(s_t, a_t) \in \mathscr{S} \times \mathscr{A}$, then (8.14) converges towards an optimal value of the Q-function when $t \rightarrow +\infty$. That is, if all the 2-tuples $(s_t, a_t) \in \mathscr{S} \times \mathscr{A}$ are visited infinitely often, the total discounted reward is guaranteed to converge to an optimal value.

- **State–action–reward–state–action (SARSA)** : this learning approach is similar to Q-learning but differs on how the Q-function is updated at each iteration. Specifically, SARSA relies on the following iterative equation:

$$Q(s_t, a_t) = (1 - \alpha_t(s_t, a_t))Q(s_t, a_t) + \alpha_t(s_t, a_t)\left[\mathscr{R}_t + \gamma Q(s_{t+1}, a_{t+1})\right] \tag{8.15}$$

  where the only difference with the Q-learning approach is that the latter updates the value of (8.14) by computing the best action such that $a^* = \arg\max_{a \in \mathscr{A}} Q_t(s, a)$. Instead, in (8.15) the algorithm utilizes the same action selection mechanism at each iteration t. That is, while Q-learning uses the optimal action a^* to update the value of (8.14), SARSA uses the same selection algorithm, e.g., ε-greedy, to compute each action a_t and directly uses it to update the Q-function in (8.15). In general, SARSA has lower-complexity if compared to Q-learning, however it often produces only near-optimal improvements at each iteration of the learning process.

8.7 Conclusions

With such massive presence of interconnected wireless nodes deployed all around us, there are still exciting yet significant security research challenges that need to be addressed in the upcoming years. In this chapter, we have provided our perspective on the issue of cross-layer wireless security, which is based on a unique mixture of machine learning and software-defined radios. Specifically, we have introduced and discussed a Learning-based Wireless Security (LeWiS) framework that provides a closed-loop approach to the problem of cross-layer wireless security by leveraging machine learning and software-defined radios. We have first provided a brief review of background notions in Sect. 8.2, followed by an in depth-discussion of the LeWiS framework and its components. We have categorized and summarized the relevant state-of-the-art research. We hope that this work will inspire fellow researchers to investigate topics pertaining to cross-layer wireless security and keep in the race for a more secure technological world.

Acknowledgements This work is based upon work supported in part by ONR grants 0014-16-1-2213 and N00014-17-1-2046, ARMY W911NF-17-1-0034, and NSF CNS-1618727.

References

1. A.A. Aburomman, M.B.I. Reaz, A novel svm-knn-pso ensemble method for intrusion detection system. Appl. Soft Comput. **38**, 360–372 (2016)
2. A.A. Ahmed, N.F. Fisal, Secure real-time routing protocol with load distribution in wireless sensor networks. Secur. Commun. Netw. **4**(8), 839–869 (2011)

3. L. Akoglu, C. Faloutsos, Anomaly, event, and fraud detection in large network datasets, in *Proceeding of the ACM International Conference on Web Search and Data Mining (WSDM)* (2013), pp. 773–774
4. I.F. Akyildiz, X. Wang, A survey on wireless mesh networks. IEEE Commun. Mag. **43**(9), S23–S30 (2005)
5. M.A. Alsheikh, S. Lin, D. Niyato, H.P. Tan, Machine learning in wireless sensor networks: algorithms, strategies, and applications. IEEE Commun. Surv. Tutorials **16**(4), 1996–2018 (2014)
6. R.A.R. Ashfaq, X.Z. Wang, J.Z. Huang, H. Abbas, Y.L. He, Fuzziness based semi-supervised learning approach for intrusion detection system. Inf. Sci. **378**, 484–497 (2017). https://doi.org/10.1016/j.ins.2016.04.019. http://www.sciencedirect.com/science/article/pii/S0020025516302547
7. D. Balfanz, D.K. Smetters, P. Stewart, H.C. Wong, Talking to strangers: authentication in ad-hoc wireless networks, in *NDSS* (2002)
8. D.P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1 (Athena Scientific, Belmont, MA, 1995)
9. A.L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutorials **18**(2), 1153–1176 (2016)
10. L. Busoniu, R. Babuska, B. De Schutter, A comprehensive survey of multiagent reinforcement learning. IEEE Trans. Syst. Man Cybern. C **38**(2), 156–172 (2008)
11. L. Buşoniu, B. De Schutter, R. Babuška, Approximate dynamic programming and reinforcement learning, in *Interactive Collaborative Information Systems* (Springer, New York, 2010), pp. 3–44
12. A. Cassola, W.K. Robertson, E. Kirda, G. Noubir, A practical, targeted, and stealthy attack against wpa enterprise authentication, in *NDSS* (2013)
13. X. Chen, K. Makki, K. Yen, N. Pissinou, Sensor network security: a survey. IEEE Commun. Surv. Tutorials **11**(2), 52–73 (2009)
14. M. Chiang, Balancing transport and physical layers in wireless multihop networks: jointly optimal congestion control and power control. IEEE J. Sel. Areas Commun. **23**(1), 104–116 (2005)
15. T.C. Clancy, Efficient ofdm denial: pilot jamming and pilot nulling, in *IEEE International Conference on Communications (ICC)* (IEEE, Kyoto, 2011), pp. 1–5
16. C. Clancy, J. Hecker, E. Stuntebeck, T. O'Shea, Applications of machine learning to cognitive radio networks. IEEE Wirel. Commun. **14**(4), 47–52 (2007)
17. S. Convery, *Network Security Architectures* (Pearson Education, Chennai, 2004)
18. M. Crawford, T.M. Khoshgoftaar, J.D. Prusa, A.N. Richter, H.A. Najada, Survey of review spam detection using machine learning techniques. J. Big Data **2**(1), 2–23 (2015)
19. M.L. Das, Two-factor user authentication in wireless sensor networks. IEEE Trans. Wirel. Commun. **8**(3), 1086–1090 (2009)
20. T.R. Dean, A.J. Goldsmith, Physical-layer cryptography through massive mimo. IEEE Trans. Inf. Theory **63**(8), 5419–5436 (2017). https://doi.org/10.1109/TIT.2017.2715187
21. L. Deng, G. Hinton, B. Kingsbury, New types of deep neural network learning for speech recognition and related applications: an overview, in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2013), pp. 8599–8603
22. L. Deng, D. Yu et al., Deep learning: methods and applications. Found. Trends Signal Process. **7**(3–4), 197–387 (2014)
23. L. Ding, T. Melodia, S. Batalama, J. Matyjas, M. Medley, Cross-layer routing and dynamic spectrum allocation in cognitive radio ad hoc networks. IEEE Trans. Veh. Technol. **59**, 1969–1979 (2010)
24. S. D'Oro, L. Galluccio, G. Morabito, S. Palazzo, Efficiency analysis of jamming-based countermeasures against malicious timing channel in tactical communications, in *2013 IEEE International Conference on Communications (ICC)* (2013), pp. 4020–4024. https://doi.org/10.1109/ICC.2013.6655188

25. S. D'Oro, L. Galluccio, G. Morabito, S. Palazzo, L. Chen, F. Martignon, Defeating jamming with the power of silence: a game-theoretic analysis. IEEE Trans. Wirel. Commun. **14**(5), 2337–2352 (2015)
26. S. D'Oro, E. Ekici, S. Palazzo, Optimal power allocation and scheduling under jamming attacks. IEEE/ACM Trans. Netw. **25**(3), 1310–1323 (2017). https://doi.org/10.1109/TNET.2016.2622002
27. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, 2nd edn. (Wiley, New York, 2000)
28. E.F. Flushing, J. Nagi, G.A. Di Caro, A mobility-assisted protocol for supervised learning of link quality estimates in wireless networks, in *International Conference on Computing, Networking and Communications (ICNC)* (IEEE, Maui, 2012), pp. 137–143
29. A.B. Gershman, U. Nickel, J.F. Bohme, Adaptive beamforming algorithms with robustness against jammer motion. IEEE Trans. Signal Process. **45**(7), 1878–1885 (1997)
30. P.K. Gopala, L. Lai, H. El Gamal, On the secrecy capacity of fading channels. IEEE Trans. Inf. Theory **54**(10), 4687–4698 (2008)
31. K. Hasan, S. Shetty, T. Oyedare, Cross layer attacks on gsm mobile networks using software defined radios, in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)* (2017), pp. 357–360. https://doi.org/10.1109/CCNC.2017.7983134
32. W. Hu, W. Hu, S. Maybank, Adaboost-based algorithm for network intrusion detection. IEEE Trans. Syst. Man Cybern. B Cybern. **38**(2), 577–583 (2008)
33. J. Huang, A.L. Swindlehurst, Cooperative jamming for secure communications in mimo relay networks. IEEE Trans. Signal Process. **59**(10), 4871–4884 (2011)
34. J.Y. Huang, I.E. Liao, Y.F. Chung, K.T. Chen, Shielding wireless sensor network using Markovian intrusion detection system with attack pattern mining. Inf. Sci. **231**, 32–44 (2013). https://doi.org/10.1016/j.ins.2011.03.014. http://www.sciencedirect.com/science/article/pii/S0020025511001435. Data Mining for Information Security
35. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in *Proceedings of the ACM International Conference on Multimedia* (2014), pp. 675–678
36. I. Jirón, I. Soto, R. Carrasco, N. Becerra, Hyperelliptic curves encryption combined with block codes for Gaussian channel. Int. J. Commun. Syst. **19**(7), 809–830 (2006)
37. J.F.C. Joseph, B.S. Lee, A. Das, B.C. Seet, Cross-layer detection of sinking behavior in wireless ad hoc networks using svm and fda. IEEE Trans. Dependable Secure Comput. **8**(2), 233–245 (2011)
38. L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
39. C. Karlof, D. Wagner, Secure routing in wireless sensor networks: attacks and countermeasures. Ad Hoc Netw. **1**(2–3), 293–315 (2003)
40. C. Karlof, N. Sastry, D. Wagner, Tinysec: a link layer security architecture for wireless sensor networks, in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems* (ACM, New York, 2004), pp. 162–175
41. Kdd cup 1999 data (1999). http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
42. R.A. Kemmerer, G. Vigna, Intrusion detection: a brief history and overview. Computer **35**(4), 27–30 (2002). https://doi.org/10.1109/MC.2002.1012428
43. D. Kreutz, F.M.W. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015). https://doi.org/10.1109/JPROC.2014.2371999
44. I. Krikidis, J.S. Thompson, S. McLaughlin, Relay selection for secure cooperative networks with jamming. IEEE Trans. Wirel. Commun. **8**(10), 5003–5011 (2009)
45. P. Laskov, P. Düssel, C. Schäfer, K. Rieck, Learning intrusion detection: supervised or unsupervised?, in *Image Analysis and Processing – ICIAP 2005*, ed. by F. Roli, S. Vitulano (Springer, Berlin/Heidelberg, 2005), pp. 50–57
46. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436 (2015)
47. K. Lee, J. Kim, K.H. Kwon, Y. Han, S. Kim, Ddos attack detection method using cluster analysis. Exp. Syst. Appl. **34**(3), 1659–1665 (2008)

48. Y. Li, L. Guo, An active learning based tcm-knn algorithm for supervised network intrusion detection. Comput. Secur. **26**(7), 459–467 (2007)
49. M. Li, I. Koutsopoulos, R. Poovendran, Optimal jamming attacks and network defense policies in wireless sensor networks, in *26th IEEE International Conference on Computer Communications. INFOCOM 2007* (IEEE, Alaska, 2007), pp. 1307–1315
50. W. Li, P. Yi, Y. Wu, L. Pan, J. Li, A new intrusion detection system based on knn classification algorithm in wireless sensor network. J. Elect. Comput. Eng. **2014**, Article ID 240217 (2014)
51. X. Lin, N.B. Shroff, R. Srikant, A tutorial on cross-layer optimization in wireless networks. IEEE J. Sel. Areas Commun. **24**, 1452–1463 (2006)
52. W.C. Lin, S.W. Ke, C.F. Tsai, Cann: an intrusion detection system based on combining cluster centers and nearest neighbors. Knowl. Based Syst. **78**, 13–21 (2015)
53. G. Liu, Z. Yi, S. Yang, A hierarchical intrusion detection model based on the pca neural networks. Neurocomputing **70**(7), 1561–1568 (2007). https://doi.org/10.1016/j.neucom.2006.10.146. http://www.sciencedirect.com/science/article/pii/S0925231206004644. Advances in Computational Intelligence and Learning
54. S. Liu, Y. Hong, E. Viterbo, Unshared secret key cryptography. IEEE Trans. Wirel. Commun. **13**(12), 6670–6683 (2014)
55. J. Lubacz, W. Mazurczyk, K. Szczypiorski, Principles and overview of network steganography. IEEE Commun. Mag. **52**(5), 225–229 (2014)
56. W. Mao, *Modern Cryptography: Theory and Practice* (Prentice Hall, Upper Saddle River, 2003)
57. D. Martins, H. Guyennet, Steganography in mac layers of 802.15. 4 protocol for securing wireless sensor networks, in *2010 International Conference on Multimedia Information Networking and Security (MINES)* (IEEE, Nanjing, 2010), pp. 824–828
58. T. Melodia, H. Kulhandjian, L.C. Kuo, E. Demirors, *Advances in Underwater Acoustic Networking* (Wiley, New York, 2013), pp. 804–852. https://doi.org/10.1002/9781118511305.ch23. http://dx.doi.org/10.1002/9781118511305.ch23
59. R.F. Molanes, J.J. Rodríguez-Andina, J. Fariña, Performance characterization and design guidelines for efficient processor - fpga communication in cyclone v fpsocs. IEEE Trans. Ind. Electron. **65**(5), 4368–4377 (2018). https://doi.org/10.1109/TIE.2017.2766581
60. L. Mucchi, L.S. Ronga, E. Del Re, A novel approach for physical layer cryptography in wireless networks. Wirel. Pers. Commun. **53**(3), 329–347 (2010)
61. L. Mucchi, L.S. Ronga, E. Del Re, Physical layer cryptography and cognitive networks, in *Trustworthy Internet* (Springer, New York, 2011), pp. 75–91
62. A. Mukherjee, S.A.A. Fakoorian, J. Huang, A.L. Swindlehurst, Principles of physical layer security in multiuser wireless networks: a survey. IEEE Commun. Surv. Tutorials **16**(3), 1550–1573 (2014)
63. M.D.V. Pena, J.J. Rodriguez-Andina, M. Manic, The internet of things: the role of reconfigurable platforms. IEEE Ind. Electron. Mag. **11**(3), 6–19 (2017). https://doi.org/10.1109/MIE.2017.2724579
64. S. Pudlewski, N. Cen, Z. Guan, T. Melodia, Video transmission over lossy wireless networks: a cross-layer perspective. IEEE J. Sel. Top. Signal Process. **9**, 6–22 (2015)
65. O. Puñal, I. Aktaş, C.J. Schnelke, G. Abidin, K. Wehrle, J. Gross, Machine learning-based jamming detection for ieee 802.11: design and experimental evaluation, in *2014 IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (IEEE, Sydney, 2014), pp. 1–10
66. E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, vol. 1 (Addison-Wesley, Reading, 2001)
67. A. Saied, R.E. Overill, T. Radzik, Detection of known and unknown ddos attacks using artificial neural networks. Neurocomputing **172**, 385–393 (2016)
68. A.L. Samuel, Some studies in machine learning using the game of checkers. IBM J. Res. Dev. **44**(1–2), 206–226 (2000)
69. K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, E.M. Belding-Royer, A secure routing protocol for ad hoc networks, in *10th IEEE International Conference on Network Protocols, 2002. Proceedings* (IEEE, Paris, 2002), pp. 78–87

70. N. Sastry, D. Wagner, Security considerations for ieee 802.15. 4 networks, in *Proceedings of the 3rd ACM workshop on Wireless Security* (ACM, New York, 2004), pp. 32–42
71. C. Shahriar, S. Sodagari, T.C. Clancy, Performance of pilot jamming on mimo channels with imperfect synchronization, in *2012 IEEE International Conference on Communications (ICC)*, pp. 898–902 (IEEE, Ottawa, 2012)
72. S. Shamshirband, A. Patel, N.B. Anuar, M.L.M. Kiah, A. Abraham, Cooperative game theoretic approach using fuzzy q-learning for detecting and preventing intrusions in wireless sensor networks. Eng. Appl. Artif. Intell. **32**, 228–241 (2014)
73. S.S.S. Sindhu, S. Geetha, A. Kannan, Decision tree based light weight intrusion detection using a wrapper approach. Exp. Syst. Appl. **39**(1), 129–141 (2012). https://doi.org/10.1016/j.eswa.2011.06.013. http://www.sciencedirect.com/science/article/pii/S0957417411009080
74. K.J. Singh, D.S. Kapoor, Create your own internet of things: a survey of IoT platforms. IEEE Consum. Electron. Mag. **6**(2), 57–68 (2017)
75. R. Sommer, V. Paxson, Outside the closed world: on using machine learning for network intrusion detection, in *2010 IEEE Symposium on Security and Privacy* (2010), pp. 305–316
76. G. Stein, B. Chen, A.S. Wu, K.A. Hua, Decision tree classifier for network intrusion detection with ga-based feature selection, in *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2, ACM-SE 43* (ACM, New York, 2005), pp. 136–141. https://doi.org/10.1145/1167253.1167288. http://doi.acm.org/10.1145/1167253.1167288
77. M. Strasser, B. Danev, S. Čapkun, Detection of reactive jamming in sensor networks. ACM Trans. Sens. Netw. **7**(2), 16:1–16:29 (2010)
78. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*, vol. 1 (MIT Press, Cambridge, 1998)
79. K. Szczypiorski, Hiccups: hidden communication system for corrupted networks, in *International Multi-Conference on Advanced Computer Systems* (2003), pp. 31–40
80. A. Testolin, M. Zanforlin, M.D.F. De Grazia, D. Munaretto, A. Zanella, M. Zorzi, M. Zorzi, A machine learning approach to qoe-based video admission control and resource allocation in wireless systems, in *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)* (IEEE, Piran, 2014), pp. 31–38
81. G. Thamilarasu, A. Balasubramanian, S. Mishra, R. Sridhar, A cross-layer based intrusion detection approach for wireless ad hoc networks, in *Proceedings of IEEE International Conference on Mobile Adhoc and Sensor Systems*, Washington (2005)
82. A.D. Tijsma, M.M. Drugan, M.A. Wiering, Comparing exploration strategies for q-learning in random stochastic mazes, in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (2016), pp. 1–8. https://doi.org/10.1109/SSCI.2016.7849366
83. O. Ureten, N. Serinken, Wireless security through rf fingerprinting. Can. J. Elect. Comput. Eng. **32**(1), 27–33 (2007)
84. S. Vadlamani, B. Eksioglu, H. Medal, A. Nandi, Jamming attacks on wireless networks: a taxonomic survey. Int. J. Prod. Econ. **172**, 76–94 (2016)
85. J.P. Walters, Z. Liang, W. Shi, V. Chaudhary, Wireless sensor network security: a survey, in *Security in Distributed, Grid, Mobile, and Pervasive Computing*, vol. 1 (Auerbach, Boston, 2007), p. 367
86. Y. Wang, M. Martonosi, L.S. Peh, Predicting link quality using supervised learning in wireless sensor networks. ACM SIGMOBILE Mob. Comput. Commun. Rev. **11**(3), 71–83 (2007)
87. X. Wang, J.S. Wong, F. Stanley, S. Basu, Cross-layer based anomaly detection in wireless mesh networks, in *Proceedings of International Symposium on Applications and the Internet*, Bellevue (2009), pp. 9–15
88. S.S. Wang, K.Q. Yan, S.C. Wang, C.W. Liu, An integrated intrusion detection system for cluster-based wireless sensor networks. Exp. Syst. Appl. **38**(12), 15234–15243 (2011)
89. X. Wang, M. Tao, J. Mo, Y. Xu, Power and subcarrier allocation for physical-layer security in ofdma-based broadband wireless networks. IEEE Trans. Inf. Forensics Secur. **6**(3), 693–702 (2011)
90. X. Wang, Q.Z. Sheng, X.S. Fang, L. Yao, X. Xu, X. Li, An integrated Bayesian approach for effective multi-truth discovery, in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (ACM, New York, 2015), pp. 493–502

91. C.J. Watkins, P. Dayan, Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)
92. D.J. White, *Markov Decision Processes* (Wiley, New York, 1993)
93. M. Wilhelm, I. Martinovic, J.B. Schmitt, V. Lenders, Short paper: reactive jamming in wireless networks: how realistic is the threat?, in *Proceedings of the Fourth ACM Conference on Wireless Network Security* (ACM, New York, 2011), pp. 47–52
94. M. Wilhelm, I. Martinovic, J.B. Schmitt, V, Lenders, Wifire: a firewall for wireless networks, in *ACM SIGCOMM Computer Communication Review*, vol. 41 (ACM, New York, 2011), pp. 456–457
95. G.J. Woeginger, Exact algorithms for np-hard problems: a survey, in *Combinatorial Optimization – Eureka, You Shrink!* (Springer, New York, 2003), pp. 185–207
96. B. Wu, J. Chen, J. Wu, M. Cardei, A survey of attacks and countermeasures in mobile ad hoc networks, in *Wireless Network Security* (Springer, New York, 2007), pp. 103–135
97. Y. Wu, B. Wang, K.R. Liu, T.C. Clancy, Anti-jamming games in multi-channel cognitive radio networks. IEEE J. Sel. Areas Commun. **30**(1), 4–15 (2012)
98. F. Wu, R. Zhang, L.L. Yang, W. Wang, Transmitter precoding-aided spatial modulation for secrecy communications. IEEE Trans. Veh. Technol. **65**(1), 467–471 (2016)
99. A.M. Wyglinski, D.P. Orofino, M.N. Ettus, T.W. Rondeau, Revolutionizing software defined radio: case studies in hardware, software, and education. IEEE Commun. Mag. **54**(1), 68–75 (2016)
100. Z. Xi, L. Li, G. Shi, S. Wang, A comparative study of encryption algorithms in wireless sensor network, in *Wireless Communications, Networking and Applications*, ed. by Q.A. Zeng (Springer, New Delhi, 2016), pp. 1087–1097
101. L. Xiao, L. Greenstein, N. Mandayam, W. Trappe, Fingerprints in the ether: using the physical layer for wireless authentication, in *IEEE International Conference on Communications, 2007. ICC'07* (IEEE, Glasgow, 2007), pp. 4646–4651
102. Q. Yan, H. Zeng, T. Jiang, M. Li, W. Lou, Y.T. Hou, Mimo-based jamming resilient communication in wireless networks, in *2014 Proceedings IEEE INFOCOM* (IEEE, Toronto, 2014), pp. 2697–2706
103. Z. Yu, J.J. Tsai, A framework of machine learning based intrusion detection for wireless sensor networks, in *IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing, 2008. SUTC'08* (IEEE, Taichung, 2008), pp. 272–279
104. L. Zhang, T. Melodia, Hammer and anvil: the threat of a cross-layer jamming-aided data control attack in multihop wireless networks, in *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, Florence (2015), pp. 361–369
105. L. Zhang, F. Restuccia, T. Melodia, S.M. Pudlewski, Learning to detect and mitigate cross-layer attacks in wireless networks: framework and applications, in *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, Las Vegas (2017), pp. 361–369
106. L. Zhang, F. Restuccia, T. Melodia, S. Pudlewski, Taming cross-layer attacks in wireless networks: a Bayesian learning approach. IEEE Trans. Mobile Comput. (2018). https://doi.org/10.1109/TMC.2018.2864155
107. J. Zheng, A. Jamalipour, *Wireless Sensor Networks: A Networking Perspective* (Wiley, New York, 2009)
108. P. Zhou, Y. Chang, J.A. Copeland, Reinforcement learning for repeated power control game in cognitive radio networks. IEEE J. Sel. Areas Commun. **30**(1), 54–69 (2012)
109. L. Zhou, D. Wu, B. Zheng, M. Guizani, Joint physical-application layer security for wireless multimedia delivery. IEEE Commun. Mag. **52**(3), 66–72 (2014). https://doi.org/10.1109/MCOM.2014.6766087
110. Y. Zou, J. Zhu, X. Wang, L. Hanzo, A survey on wireless security: technical challenges, recent advances, and future trends. Proc. IEEE **104**, 1727–1765 (2016)
111. F. Restuccia, T. Melodia, Big data goes small: real-time spectrum-driven embedded wireless networking through deep learning in the RF loop, in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, Paris (May 2019)

112. J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, T. Melodia, Machine learning for wireless communications in the Internet of things: a comprehensive survey. Preprint. arXiv:1901.07947
113. S. D'Oro, F. Restuccia, T. Melodia, Hiding data in plain sight: undetectable wireless communications through pseudo-noise asymmetric shift keying, in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, Paris (May 2019)
114. F. Restuccia, S. D'Oro, T. Melodia, Securing the internet of things in the age of machine learning and software-defined networking. IEEE Internet Things J. **5**(6) (2018)
115. F. Restuccia, N. Ghosh, S. Bhattacharjee, S.K. Das, T. Melodia, Quality of information in mobile crowd sensing: survey and research challenges. ACM Trans. Sensor Netw. **13**(4), 34 (2017)