Intrusion Detection in Wi-Fi Networks by Modular and Optimized Ensemble of Classifiers

Giuseppe Granato[a], Alessio Martino[b], Luca Baldini[c] and Antonello Rizzi[d]
Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza", via Eudossiana 18, 00184 Rome, Italy

Keywords: Information Granulation, Data Clustering, Supervised Learning, Genetic Algorithms, Malicious Traffic Detection, Network Intrusion Detection Systems.

Abstract: With the breakthrough of pervasive advanced networking infrastructures and paradigms such as 5G and IoT, cybersecurity became an active and crucial field in the last years. Furthermore, machine learning techniques are gaining more and more attention as prospective tools for mining of (possibly malicious) packet traces and automatic synthesis of network intrusion detection systems. In this work, we propose a modular ensemble of classifiers for spotting malicious attacks on Wi-Fi networks. Each classifier in the ensemble is tailored to characterize a given attack class and is individually optimized by means of a genetic algorithm wrapper with the dual goal of hyper-parameters tuning and retaining only relevant features for a specific attack class. Our approach also considers a novel false alarm management procedure thanks to a proper reliability measure formulation. The proposed system has been tested on the well-known AWID dataset, showing performances comparable with other state of the art works both in terms of accuracy and knowledge discovery capabilities. Our system is also characterized by a modular design of the classification model, allowing to include new possible attack classes in an efficient way.

1 INTRODUCTION

Recent developments in wireless networking technology have been key elements in the evolution of future smart environments, and in the last few years technology has deeply evolved to fulfil needs in different areas. Starting from enterprise wireless networks, considering also Internet of Thing (IoT) networks, the introduction of new smart services implied the birth of new attack scenarios. Furthermore, in this kind of communication environments, the complexity of the adopted protocols and the high capacity of the network infrastructure bring the need to develop and employ new processing strategies for this kind of Big Data. At this purpose, this field has been deeply studied using computational intelligence approaches over the last two decades (Sperotto et al., 2010; Bhuyan et al., 2014) following the new technological innovation in this area, from IoT (Roux et al., 2018) to Software Defined Networking (Abhilash and Divyansh, 2018). The importance gained by the IoT cybersecurity research field has furthermore highlighted the importance of building Network Intrusion Detection Systems (NIDSs) capable of working in an environment made of mobile heterogeneous devices, connected by the IEEE 802.11 standard (Wi-Fi). So, extending the focus also on the Physical and Medium Access Control (MAC) levels, rather than just network, transport or application levels, is needed to fulfil the security gap in IoT and industrial sensors networks (Roux et al., 2018; Anton et al., 2019). For this purpose, in (Kolias et al., 2016) an extensive study of possible Wi-Fi attacks in a real Small Office Home Office (SOHO) environment has been carried out, which included multiple workstations and smart devices. The major contribution proposed in (Kolias et al., 2016) is the publication of the Aegean Wi-Fi Intrusion Detection (AWID) dataset, able to work as an interesting test bed for new processing strategies for network anomaly and intrusion detection.

In this work, we further study and develop computational intelligence tools starting from a previous recent work (Rizzi et al., 2020), in order to provide advanced approaches for network attack classification

[a] https://orcid.org/0000-0001-8014-9152
[b] https://orcid.org/0000-0003-1730-5436
[c] https://orcid.org/0000-0003-4391-2598
[d] https://orcid.org/0000-0001-8244-0015

Granato, G., Martino, A., Baldini, L. and Rizzi, A.
Intrusion Detection in Wi-Fi Networks by Modular and Optimized Ensemble of Classifiers.
DOI: 10.5220/0010109604120422
In *Proceedings of the 12th International Joint Conference on Computational Intelligence (IJCCI 2020)*, pages 412-422
ISBN: 978-989-758-475-6

and automatic knowledge discovery. The key points that we target in this work are:

- an improved modular classification system able to be automatically adapted to detect multiple kind of attacks, potentially not exclusively in Wi-Fi networks;
- an improved processing methodology with further automatic knowledge discovery capabilities at the training stage;
- a further study on different approaches to detect and classify attacks based on *single frame analysis*, hopefully to employ relatively cheap hardware to gain maximum operational advantage.

Our proposed processing architecture is based on an array of simple two-class classifiers, each one specialized in discriminating frames specifically forged to be part of a specific attack against all other types of frames. We further enhance the knowledge discovery capabilities of the training phase by adopting a genetic meta-heuristic procedure that not only selects relevant features, but also tunes the hyper-parameters of the employed clustering procedure and classification algorithm in order to maximize the classification accuracy and the knowledge discovery capabilities. Lastly, we show that a remarkable accuracy can be obtained in spotting malicious frames, while maintaining a low false alarm rate. In particular, out of 14 attack classes, the proposed strategy is able to achieve an accuracy level greater than 90% for 10 of them and 99% for 7 of them.

The rest of the paper is organized as follows. After reviewing the related literature in Section 2, we present the main characteristics of the AWID dataset and outline the attacks in Section 3, along with the pre-processing stage and the definition of the dissimilarity measure adopted to quantify the dissimilarity between data packets. An account of the machine learning algorithms used for our system is given in Section 4. The design of Wi-Fi attacks classifier is outlined in Section 5. The experimental results are presented in Section 6. Finally, conclusions and future works are drawn in Section 7.

2 RELATED WORKS

The Big Data problem defined in the previous Section has been deeply studied using machine learning techniques as a mean to develop automatic anomaly detection and attack classification tools. One of the key advantages of machine learning techniques over traditional approaches is that they do not demand pre-made signatures of attack frames (Bhuyan et al., 2014). In particular, in the field of IoT and enterprise Wi-Fi networks, multiple approaches have been studied so far. Starting with the work of Kolias et al. (Kolias et al., 2016), a deep study of Wi-Fi attacks in WEP protected networks has been carried out in order to correctly build a reference dataset to study different approaches to the intrusion detection problem. The authors present the first attempt to perform network traffic classification over the AWID dataset using machine learning techniques provided by the open source software WEKA (Frank et al., 2016). Furthermore, authors provide a first analysis of key features required for the classification problem.

In (Benzaïd et al., 2016) network traffic has been analyzed at MAC level using artificial neural networks, suitably trained to detect address spoofing attacks.

In (Guennoun et al., 2008; El-Khatib, 2010) an approach based on *k*-means clustering and multilayer perceptrons has been studied, used jointly with information gain methods, achieving accuracy near 90%. The dataset used in those works is not public.

Multiple machine learning algorithms have been employed in (Agarwal et al., 2015) in order to detect de-authentication Denial of Service attacks.

Particular focus has been given to the `evil_twin` attack and fake Access Point (AP) detection in (Takahashi et al., 2010) and (Lanze et al., 2014), in order to find anomalous network traffic using fingerprinting techniques and to distinguish (real) hardware APs from (fake) software ones.

In (Thing, 2017) a deep learning approach for multi-class classification has been considered in order to detect macro-classes of network traffic (legitimate traffic, *flooding*, *injection* and *impersonation* attacks), along with feature self-learning strategies by means of a deep learning approach based on a stacked auto-encoder with different activation functions.

In (Aminanto and Kim, 2017), the authors focus on improving feature selection and detection of the *impersonation* attacks by using a deep learning approach based on a stacked auto-encoder algorithm on the AWID dataset. They exploit artificial neural networks to perform feature selection adopting a manual threshold applied to the first hidden layer of an artificial neural network and a stacked auto-encoder approach to classify patterns.

In (Kolias et al., 2017), ant colony optimization has been employed on a pre-processed AWID dataset, aggregating features to preserve privacy in a central node and find new ways to detect attacks. The authors perform off-line traffic analysis by adopting a MAC cumulative statistics approach studying frames in pre-defined time windows. Effectiveness of the developed algorithm is demonstrated in particular by the interesting IF-THEN rule-based interpretability point of view, for which has been described how rich of information could be frames' time windows to find malicious frames.

In (Qin et al., 2018), the authors proposed an intrusion detection study based on a custom manual pre-processing scheme based on the calculation of linear dependencies between features, with a features selection stage using a two dimensional data cleaning approach. Finally, Support Vector Machines with Gaussian radial basis function kernel are employed for classification. This classification scheme realizes an ensemble of binary classifiers in order to solve the multi-class classification problem. In this work, the authors used the AWID dataset by adopting the same 4-way labelling already used in (Thing, 2017).

Combined use of Kernel Density Estimation and Hidden Markov Models through a tandem queueing network model has been exploited in (Sethuraman et al., 2019). Authors pre-processed a subset of the AWID dataset, selecting features using a probabilistic approach in order to obtain a network flows set of patterns. The trained model is capable of obtaining interesting performance. The interpretability of computational intelligence models is the core of this work, which stems from a previous study (Rizzi et al., 2020). In the latter, we also exploited patterns' features extracted from a *single* MAC frame in order to check its maliciousness, and associate it to specific attack strategies. In this work, we further propose:

- a revised optimization process based on a new fitness function which jointly considers a suitably defined penalty function and the Youden's J statistic (also, *informedness*) as performance values;
- a new genetic code for the optimization procedure, which includes the hyper-parameters for both clustering and classification stages in order to select, for each attack class, the best training set resolution and the best setup parameters;
- a second lightweight genetic algorithm able to adapt thresholds for false alarm management.

3 DATASET DESCRIPTION

In this work we make use of the AWID dataset to check performance and evaluate the validity of the proposed system. This dataset has been built by capturing heterogeneous Wi-Fi traffic in a SOHO network environment realized with multiple devices (workstations, notebooks, smartphones, smart TVs), along with a monitor node that passively captures network traffic and an attack node equipped with Kali Linux. The networking environment where attacks are carried out consists of infra-structured Wi-Fi networks, i.e., Wi-Fi networks where stations communicate with an AP. The AP issues periodically a so-called *beacon* message, announcing itself and a number of parameters and attributes useful for coordinating the access to the wireless channel. The whole network is protected by the WEP security protocol. WEP has been marked as obsolete due to various security flaws, and replaced by the protocols specified in the IEEE 802.11i standard. However, WEP has been chosen for the AWID dataset to simplify the data set acquisition and the construction of the ground truth information for intrusion detection system experimentation. Moreover, it is interesting to include attacks on WEP to test the ability of the machine learning traffic analyzer to detect those attacks as "anomalies" with respect to plain network traffic. Indeed, some of the attacks that have been carried out in the test bed network can be used against WPA2 networks as well (see Section 3.2). Regular traffic (that is, not affected by any attack) has been generated by common applications, such as web browsing, file transfer, audio/video streaming. Attacks have been realized by means of the state-of-the-art *aircrack-ng* suite[1], MDK3[2] and other ad-hoc tools. Network traffic belonging to each attack has been labeled with the related class.

The dataset used in this work is composed of three parts:

- a single traffic trace which contains examples of all attack classes, that we take to define the raw training set S'_{tr};
- twelve trace files containing legitimate ('normal') traffic, that we use as a first test set (S_{ts1});
- twelve trace files containing mixed normal and attack traffic, that we use as a second test set, (S_{ts2}).

The rationale behind this choice consists in the possibility of highlighting classifiers' performance in two scenarios: one where no attack is in progress (only normal traffic), against a second case where a mixed situation is realized, with normal traffic interleaved with attack traffic, belonging to different attack types. A concise description of the attack macro-classes in the AWID dataset is outlined in Sections 3.1–3.3[3], with Table 1 summarizing the patterns distribution for each class within the three different sets, whereas in Sections 3.4–3.5 we describe the pre-processing phase and the adopted dissimilarity measure between MAC frames.

[1]https://www.aircrack-ng.org

[2]https://tools.kali.org/wireless-attacks/mdk3

[3]Further details can be found in (Kolias et al., 2016).

3.1 Flooding

This kind of attacks exploits the lack of authentication and integrity check in control and management frames (see (IEEE, 2016) for the detailed definition of frame classes) that leads to frames with multiple malleable fields. The strategy used to modify frame's fields characterize the specific attack and, depending on the toolchain used, could lead to recognisable values, for instance in terms of *sequence number*, *reason code* or *received signal strength*. Eight classes belong to the flooding family: `beacon`, `de-authentication`, `disassociation`, `amok`, `power_saving`, `probe_request`, `rts` and `cts`.

3.2 Frame Injection

This family of attacks aims to forge frames exploiting vulnerabilities of the WEP security protocol. Forged frames could be used, for instance, to solicit responses by victim stations in order to collect cryptographic material exploitable to recover the keystream, or using the AP as an oracle and decrypt frames. This kind of attacks are made possible thanks to security flaws of WEP, which basically are linked to the malleability of the enciphering and a lack of a real message authentication code. `Arp`, `chop_chop` and `fragmentation` belong to this family.

3.3 Impersonation

Impersonation attacks make use of multiple of the previously described vulnerabilities in order to setup fake APs as a starting point for more complex attacks. For instance, these attacks can adopt techniques typical of `arp` and `fragmentation` attacks to collect cryptographic material. This family sees `evil_twin`, `cafe_latte` and `hirte` attacks.

3.4 Pre-Processing

In this work we adopted a subset of the AWID dataset in order to: suitably train the classification system (S'_{tr}), check its capabilities in term of false alarms on a dedicated set made entirely of normal network traffic (S_{ts1}), and verify the accuracy on a mixed attacks case (S_{ts2}). The training set has been further split in order to obtain a validation set S_{val} useful for optimizing performance indices during the training phase. The applied pre-processing phase has been split up in two parts: feature engineering and feature normalization. The former aims to select the most useful features from the raw data, whereas the latter is employed to avoid implicit weighting phenomena. Starting from the 156 attributes[4] (e.g., MAC addresses, header flags, timestamp, ...) composing each pattern in the AWID dataset, we have carefully chosen a subset of 25, considered relevant for the purpose of network intrusion detection following the same rationale behind our previous work (Rizzi et al., 2020). This starting set of features are summarized in Table 2, and are composed by either numerical, nominal or boolean types. All considered numerical features range between 0 and a maximum value that can be derived from the IEEE 802.11 specifications (IEEE, 2016), hence each numerical feature f is normalized as $f^{(\text{norm})} = f/f_{\max}$.

Table 1: Patterns per class distribution in the three sets.

Class name	S'_{tr}	S_{ts1}	S_{ts2}
normal	530785	35158851	47325477
amok	477	0	3856
arp	13644	0	500823
beacon	599	0	5498
cafe_latte	379	0	16719
chopchop	2871	0	22879
cts	1759	0	38359
deauthentication	4445	0	33870
disassociation	84	0	34871
evil_twin	611	0	27045
fragmentation	167	0	240
hirte	19089	0	433750
power_saving	165	0	13551
probe_request	369	0	10981
rts	199	0	13536
Total	565643	35158851	48481455

3.5 Dissimilarity Measure

Designing an appropriate dissimilarity measure for the problem and data at hand is a key facet for the success of any pattern recognition system. As for Table 2, we consider patterns made up of numerical (integer- or real-valued), discrete nominal and boolean features. Numerical features are assumed to be normalized in $[0,1]$. In this work, we adopt the following dissimilarity measure, ad-hoc tailored to the heterogeneous pattern structure: let $\mathbf{x}$ and $\mathbf{y}$ be two generic patterns, then if the i^{th} feature is numerical we let

$$d_i(\mathbf{x}_i, \mathbf{y}_i) = |\mathbf{x}_i - \mathbf{y}_i| \tag{1}$$

whereas if the i^{th} feature is nominal or boolean, we let

$$d_i(\mathbf{x}_i, \mathbf{y}_i) = \begin{cases} 1 & \mathbf{x}_i \neq \mathbf{y}_i \\ 0 & \mathbf{x}_i = \mathbf{y}_i \end{cases} \tag{2}$$

[4]http://icsdweb.aegean.gr/awid/

Table 2: List of considered features.

Attribute name	Data type	Description
frame.len	Numerical	Frame length
radiotap.dbm_antsignal	Numerical	Received Signal Strength
wlan.fc.type	Nominal	Frame Type
wlan.fc.subtype	Nominal	Frame Subtype
wlan.fc.ds	Nominal	Distribution System status
wlan.fc.frag	Boolean	More Fragments
wlan.fc.pwrmgt	Boolean	Power management
wlan.fc.order	Boolean	Order flag
wlan.duration	Numerical	Duration
wlan.ra	Nominal	Receiver address
wlan.da	Nominal	Destination address
wlan.ta	Nominal	Transmitter address
wlan.sa	Nominal	Source address
wlan.bssid	Nominal	BSS ID
wlan.frag	Numerical	Fragment number
wlan.seq	Numerical	Sequence number
wlan.fcs_good	Boolean	FCS correctness
wlan_mgt.fixed.listen_ival	Numerical	Listen Interval
wlan_mgt.fixed.timestamp	Numerical	Timestamp
wlan_mgt.fixed.beacon	Numerical	Beacon Interval
wlan_mgt.fixed.reason_code	Nominal	Reason code
wlan_mgt.fixed.sequence	Numerical	Starting Sequence Number
wlan.wep.iv	Nominal	Initialization Vector
wlan.wep.icv	Nominal	Integrity Check Value
data.len	Numerical	Data Length
Class label	String	Ground truth metadata

Finally, if we allow each feature to be weighted independently by means of a binary weighting vector $\mathbf{w} \in \{0,1\}^n$, the overall dissimilarity measure reads as:

$$d(\mathbf{x},\mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}_i \cdot d_i(\mathbf{x}_i,\mathbf{y}_i) \tag{3}$$

where n is the number of considered features. Consistently with the definitions given above, the dissimilarity measure takes values in range $[0,1]$.

4 ADOPTED CLASSIFICATION SYSTEM

The computational intelligence approach proposed in this work is composed by the following key components, which will be independently described in Sections 4.1–4.3:

- a *granulation strategy*: in order to reduce the cardinality of the training set while maintaining useful information, we employ a clustering algorithm that groups patterns in clusters, each one identified by its representative pattern;
- a *classification algorithm*: which maps patterns with labels according to a given strategy;
- a *genetic optimization procedure*: used to find a subset of features per class and an optimal choice of hyper-parameters for both the granulation strategy and the classification algorithm.

4.1 Information Granulation by Clustering

The training set includes redundant information, making it difficult to train algorithms on it without performance issues. At this purpose, we introduce an information granulation strategy in order to reduce the cardinality of the training data and to simplify the training phase. We define an *information granule* as a collection of entities arranged together thanks to their similarity. So, from this point of view, a cluster, seen as a set of similar patterns, is a typical example of information granule (Baldini et al., 2019). Thanks to a clustering algorithm and by representing each cluster according to a unique representative pattern, it is possible to perform an information compression to reduce the cardinality of a data set, whilst preserving most of the useful information.

To this aim, we adopted the well-known Basic Sequential Algorithmic Scheme (BSAS) (Theodoridis and Koutroumbas, 2008), a free clustering procedure where the number of clusters to find is not given a-priori, being an output of the clustering procedure itself. The BSAS algorithm depends on the scale parameter θ, by which it is possible to set the resolution at which the dataset is analysed that, in turn, is directly related to the compression ratio. Small values of θ yield a great number of small clusters, while higher values for θ return fewer and larger clusters. Basically, the BSAS algorithm scans the entire set S'_{tr} and alternatively assign each pattern to an existing cluster or to a newly initiated one, depending on whether the distance with respect to already existing clusters is greater than or less than θ. After all patterns in S'_{tr} have been scanned, a set of N_P clusters emerges. For each cluster $\mathcal{C}$, its representative is set by the Minimum Sum of Distances (MinSoD) criterion (Martino et al., 2017; Martino et al., 2019b), hence we select the pattern $y \in \mathcal{C}$ that minimizes the sum $\sum_{x\in\mathcal{C}} d(x,y)$, where $d(\cdot,\cdot)$ is the dissimilarity measure. Hereinafter, let us denote S_{tr} the compressed version of S'_{tr}.

4.2 *K*-Nearest Neighbours

The classification phase has been carried out using the *K*-Nearest Neighbours (*K*-NN) (Cover and Hart, 1967) decision rule. Despite its limitations, it is easily customizable by means of a suitably defined dissimilarity measure, possibly tailored to work with the structured data at hand, as in this case.

The *K*-NN algorithm is defined by the set of known patterns S_{tr}, an integer K and a dissimilarity measure $d(\cdot,\cdot)$. Then, given a pattern to be classified, the label associated to it is assigned by considering the most frequent labels among its K closest neighbours according to $d(\cdot,\cdot)$. Note that K-NN, in its plain version, does not include any training phase. The set of patterns used to evaluate the nearest neighbours is directly the set of patterns belonging to the (compressed) training set S_{tr}. Hence, the complexity of this decision rule is given by the cardinality of the training data (i.e., $N_P = |S_{tr}|$). In spite of its simplicity, K-NN is an effective tool, usually providing good performances, once a suitable dissimilarity measure has been defined, at the cost of using the entire training set as a classification model. In our case, it is safe to say that the use of the K-NN rule is strictly related to the clustering procedure used to granulate the training set and to compress information.

4.3 Genetic Algorithm

Patterns in the (compressed) training set are characterized by 25 features, each related to a given weight, to which we add the granulation parameter and the classification algorithm hyper-parameters. An exhaustive search in this space is obviously unfeasible and an effective way to face an automatic feature selection problem consists in adopting evolutionary optimization procedures, such as a genetic algorithm (Goldberg, 1989), with the final goal of optimizing a given *fitness function*. Each solution in the admissible domain is represented by a data structure (called *genetic code*) composed by 27 variables, summarized in Table 3. Let G denote the size of the population (i.e., number of individuals), with the first population being randomly generated. The fitness of each individual is evaluated and the population is properly sorted according to their fitness values. At each iteration, a new generation is constructed according to standard operators:

Elitism. best individuals are copied to the next generation;

Selection. s individuals are selected for mating with probability r;

Crossover. random exchange of genes between the s individuals previously selected;

Mutation. random flipping of genes;

Immigration. remaining $G - ([\alpha G] + s)$ individuals are randomly defined in the next generation.

The evolutionary optimization is stopped when the improvement of the fitness of the best individual becomes marginal. In our implementation, we set $G = 50$, $\alpha = 0.1$, $r = 0.2$, $q = 0.7$.

5 ENSEMBLE OF BINARY CLASSIFIERS FOR WI-FI ATTACKS DETECTION

5.1 Training the System

The proposed architecture is sketched in Fig. 1. The whole classification system is based on an ensemble of binary classifiers, each one specialized in detecting a given class. For this purpose, each classifier is individually optimized by means of a first genetic algorithm (Section 4.3) in order to find suitable hyper-parameters (K, θ) and suitable relevant features ($\mathbf{w}$, see Eq. (3)) for each of the problem-related classes. A second genetic algorithm will be responsible to optimize recognition thresholds to finally decide which label has to be assigned to each pattern (Section 5.2). This approach is useful in all those cases in which we have multi-class classification problems with overlapping decision regions that make patterns classification difficult. Furthermore, this allows to carry knowledge discovery on a class-aware basis, allowing to select a tailored subset of useful features for each attack class.

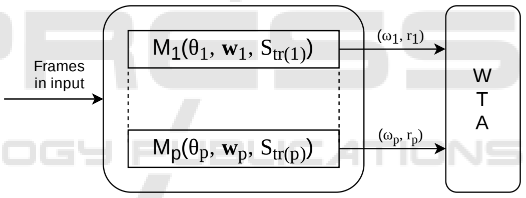
Frames in input
$M_1(\theta_1, \mathbf{w}_1, S_{tr(1)})$
$M_p(\theta_p, \mathbf{w}_p, S_{tr(p)})$
(ω_1, r_1)
(ω_p, r_p)
W T A
Figure 1: Proposed architecture: each model M_i, related to the i^{th} class, processes the input frame by considering the weighting vector $\mathbf{w}_i$ and the custom training set $S_{tr(i)}$ obtained through the granulation strategy setup with θ_i.

Table 3: Genetic code description.

Parameter	Range
Max cluster radius θ	$[0, 0.1]$
Number of neighbours K	$[1, 21]$
Feature selector $\mathbf{w}_i$	$\{0, 1\}$

The fitness function of the genetic algorithm is made of two main contributions: the *informedness* (see Eq. (4)), a summary performance index, and the *penalty* (see Eq. (5)), a correction factor useful to avoid an extreme training set compression (i.e., risk of removing useful information). The *informedness* index represents the probability of an informed decision, giving equal importance to false positive and false negative values (Powers, 2011). Different measurements that gives the same index value have the same proportion of total misclassified results. This

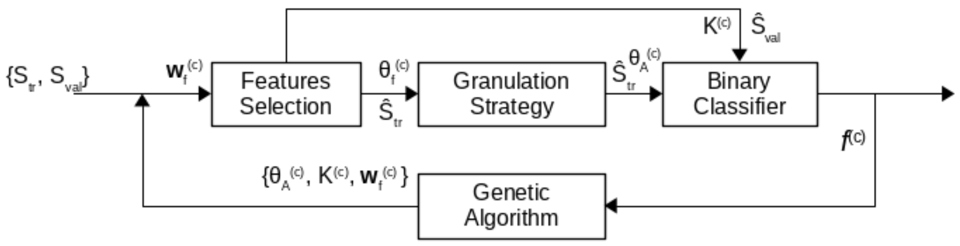

Figure 2: Proposed processing architecture for the training phase.

performance index is useful in operational contexts where we have the vast majority of patterns representing normal traffic packets, so the high false alarm rate could imply an heavy loss of network throughput. Formally, we can introduce the informedness index (Youden, 1950), normalized as in (Martino et al., 2019a; Martino et al., 2020a; Martino et al., 2020b), with the following equation:

$$J = \frac{S_e + S_p}{2}, \qquad J \in [0,1] \tag{4}$$

where S_e and S_p indicate sensitivity and specificity, respectively. The *penalty* index, as instead, has been introduced in order to effectively perform training set compression without loosing excessive information. This index will be significantly high in case the clustering algorithm returns a number of representatives for a given class lower than 2. Furthermore, the number of neighbours K must be chosen compatibly with the quantity of representatives found. So we assign a penalty value to each single class outcome of the clustering algorithm applied to the training set, the mean of that values will be subtracted from the classification performance within the fitness function for each individual of the genetic algorithm. Mathematically, the penalty function is described by the following equation:

$$P = \frac{1}{p}\sum_{i=1}^{N} e^{-(m_i(\theta_i)-1)}, \qquad P \in [0,1] \tag{5}$$

where p is the number of classes, and $m_i(\theta_i)$ is the number of clusters found for class i, which depends on the θ_i parameter for the granulation strategy. Hence the fitness function, to be maximized on S_{val}, reads as:

$$f = J(\mathbf{w}_i) - P(\theta_i), \qquad f \in [0,1] \tag{6}$$

5.2 Contentions Resolution

Each binary One-Against-All classifier belonging to the ensemble detects if a given pattern belongs to the target class, so it could happen that multiple classifiers mark with different labels the same Wi-Fi frame. In order to properly evaluate the correctness of each classifier, we adopt a Winner-Takes-All (WTA) strategy based on the reliability measure analysis of the output labels of each classifier. This analysis is performed using a simple multi-class K-NN decision rule, that processes each frame by considering the output of each classifier. So each classifier, for each pattern, returns the predicted label and the reliability measure of its prediction. Mathematically, we used the following definition as reliability measure:

$$R = \frac{\frac{1}{K}\sum_{\forall i \in \mathcal{W}}(1-d_i) - \frac{1}{K}\sum_{\forall i \in \mathcal{S}}(1-d_i) - \left(\frac{-\frac{K}{2}-1}{K}\right)}{1-\left(\frac{-\frac{K}{2}-1}{K}\right)} \tag{7}$$

where $\mathcal{W}$ and $\mathcal{S}$ are the sets of neighbours belonging to the first and second most popular class, respectively. This kind of measure:

- considers the purity of the neighbours set (i.e., the number of patterns of the winner class) and their distances d_i with respect to the test pattern;
- considers the number of patterns of the looser class and their corresponding distances from the classified pattern;
- lastly, takes values in the range $[0,1]$.

Since we are adopting a feature selection mechanism, each classifier will likely work on a different subspace of the dataset. In order to fairly compare their outputs in the WTA, we performed a further normalization of the dissimilarity measure (formerly Eq. (3)). Hence, for the j^{th} classifier, we have:

$$d_j(\mathbf{x},\mathbf{y}) = \frac{1}{n}\frac{1}{\sqrt{n_j}}\sum_{i=1}^{n}\mathbf{w}_i \cdot d_i(\mathbf{x}_i,\mathbf{y}_i) \tag{8}$$

where n_j indicates the number of selected features by the j^{th} classifier.

5.3 False Alarm Management

NIDSs' actions have an impact on the Quality of Service of the network. One of the key parameters that let us to evaluate NIDSs activity during attack prevention is the false alarm rate. An excessive number of false alarms raised by the NIDS can be harmful both in the cases of suspicious frames dropping and, from a security analyst viewpoint, might lead to handling a large volume of false alarms that need a proper counteraction, according to the security policy. In this work, we studied a solution able to reduce this problem based on a dedicated binary classifier which targets the normal class and the evaluation of a reliability measure for binary classifiers outputs (see Section 5.2). Normal network traffic classification is expected to be a difficult task, since in the same set we consider highly heterogeneous frames (for example, think about the legit management and control frames or the dynamics of the exchange of data frames between stations, depending on the kind of network layer traffic brought). At the same time, introducing an estimate of traffic normality can add an important information to use during the execution of the WTA rule. So, we basically add a new contender during this phase.

The false alarm management phase is split in three steps:

1. introduce a dedicated binary classifier which targets normal frames;
2. apply a threshold τ on the reliability measure of the output label of each classifier;
3. optimize these thresholds using a genetic optimization procedure in order to let the ensemble of classifiers to take decisions about labels.

This genetic procedure is based on the same standard principles of the previous one, yet the genetic code is defined as the vector of reliability thresholds as returned by the classifier ensemble. The fitness corresponds to the precision of each classifier.

6 TESTS AND RESULTS

At the end of the training procedure, the genetic algorithm returns, for each label, an optimal set of features and a suitable set of hyper-parameters for both clustering and the classification algorithms. Conversely, the second genetic optimization phase adapts reliability measure thresholds to improve perfomance and false alarms management. In Table 4 we summarize the genetic codes found during the genetic optimization procedures (K, θ, $\mathbf{w}$ and τ), along with other indices such as informedness, precision, best fitness and cardinality of the compressed training set, in order to let us analyze results from different perspectives. In Table 5 we have resumed results obtained using both test sets S_{ts1} and S_{ts2}.

As we can see, performance are interesting and we can make some considerations. First of all, we confirm that almost all flooding attacks are easily identifiable on a per-frame basis. This is particularly true for all those attacks based on management frames, for which we can provide a precise feature set. As shown in Table 4, we can find all key features automatically selected by this system for this type of attacks (e.g., *type*, *subtype*, *reason_code*). After all, this conclusion is not true for flooding attacks based on control frames like `rts` and `cts`. In fact, these attacks exploit frames used for MAC, so they are very simple and not identifiable without taking in consideration an analysis based on a suitably ordered set of frames. Performance indexes obtained on a mixed attack case (S_{ts2}) are confirmed also in a normal-only situation (S_{ts1}), demonstrating low false alarm rate (i.e., 1's complement of the reported accuracy) capabilities for the trained classification system.

Secondly, concerning attacks that aim to decrypt the payload or retrieve the keystream itself, results are similar to flooding attacks. When this kind of attacks become part of more complex attack vectors, classification performance drop since a flow-based analysis approach is required.

The introduction of the reliability measure threshold based on the normal One-Against-All classifier, allows us to maintain low false alarm rates, paying the price of lower classification accuracy since the rule to assign an attack is much more severe.

Lastly, impersonation attacks are the most difficult to classify due to the extremely similar behaviour to a normal scenario. The accuracy obtained in these cases is low and highly variable depending on the particular test set.

An almost common key feature automatically selected by the classification system during the training phase is the received signal power strength, which is measured by the wireless receiver of the monitor station. In this manner, the training procedure is able to take in consideration the spatial position of the transmitter station, embedded in the received signal power strength. On a single frame classification problem, this could be the best feature able to help detect flooding attacks based on `rts` and `cts` control frames, (except for the classic MAC addresses dictionary, which is not easily scalable).

We conclude this analysis by comparing, as much as possible, results obtained in this work against other recent works (see Table 6) on the same dataset, to the best of our knowledge. Since those works employ different AWID subsets, different class labels subsets and different objective functions, performance comparison is only possible to a limited extent.

Methodologically speaking, one of the most important aspects developed in our work that others do

Table 4: Optimization phase results for each classifier. Features in the rightmost column are ordered as in Table 2.

Class name	θ	$\lvert Str \rvert$	K	**Best Fitness**	J	**Precision**	τ	**Selected Features (w)**
amok	0.0029	28	5	0.8238	0.9967	0.8681	0.689	1110001010111111100110110
arp	0.0002	800	1	0.998	0.9985	1	0.934	1111111111111101110110111
beacon	0.0091	263	1	0.7751	0.997	0.9944	0.806	1101001011111111111110111
cafe_latte	0.0049	8	1	0.8126	0.994	0.0263	0.75	1111111110111111110101111
chop_chop	0.0104	800	1	0.7254	0.9534	0.9977	0.856	0010001110111111101110110
cts	0.0005	15	1	0.9393	0.9489	1	0.96	0111111111111101110010111
deauthentication	0.004	6	11	0.8169	1	0.997	0.045	1110000110001111101011110
disassociation	0.0009	22	5	0.8759	0.9999	1	0.983	0011101011111111100110111
evil_twin	0.0047	3	5	0.5054	0.5271	0.0109	0.024	1100000001001111101010110
fragmentation	0.0002	107	7	0.8280	0.9971	1	0.656	1011111110001111110010110
hirte	0.0587	51	1	0.5213	0.9861	0.9923	0.732	1101111101111110100110111
power_saving	0.0029	10	1	0.8636	0.9907	1	0.988	0100111100111110111010110
probe_request	0.0028	2	1	0.8272	0.9999	1	0.91	1101100101111111110011111
rts	0.0002	6	1	0.9983	0.9988	1	0.965	1111111111111111111111111

Table 5: Mean accuracy obtained on the two considered test sets.

Class name	S_{ts1} **Mean (Std)**	S_{ts2} **Mean (Std)**
amok	0.9999 (0.0002)	0.9999 (1.25e-07)
arp	0.9270 (0.0634)	0.9176 (0.0042)
beacon	0.9825 (0.0131)	0.9989 (4.77e-07)
cafe_latte	0.7911 (0.1092)	0.9169 (0.0012)
chop_chop	0.9895 (0.0112)	0.9899 (0.0001)
cts	0.7582 (0.171)	0.5177 (0.0031)
deauthentication	1 (0)	0.9998 (5.02e-07)
disassociation	1 (0)	0.9997 (4.98e-10)
evil_twin	1 (0)	0.9993 (1.98e-06)
fragmentation	1 (0)	0.9997 (2.66e-07)
hirte	0.3205 (0.1811)	0.7015 (2.9e-07)
power_saving	0.6966 (0.1275)	0.8689 (0.0042)
probe_request	1 (0)	0.9999 (5e-10)
rts	0.9599 (0.0576)	0.844 (0.0031)

not employ is an automatic algorithm setup and features selection strategy based on a genetic algorithm which detects relevant features in an attack-aware fashion, returning interpretable information (genetic codes). Furthermore, we employed two different subset of the AWID dataset in order to perform an explicit evaluation of our system for false alarms management. Other than numerical results, the following aspects hold: in (Thing, 2017), the overall accuracy is greater than 98%, but with some caveats considering the flooding attack class. Furthermore, they employ an embedded self-learning approach via a deep neural network for feature selection, yet do not provide an in-depth analysis. Results in (Aminanto and Kim, 2017) are rather comparable with the proposed ones. In (Kolias et al., 2017), despite achieved accuracy is 77.8%, the output of the classification algorithm includes human-readable IF-THEN rules to perform filtering activities. In our case, we do not provide explicit rules, but we find key features of each attack in order to try to understand and characterize malicious-vs-normal Wi-Fi flows. Finally, in (Qin et al., 2018), the set of interesting attributes found with their technique is similar to these found in this work. Classification accuracy of injection attacks is comparable, while impersonation and flooding attacks are recognized hardly. Overall, the advantage of the proposed approach, if compared in particular to deep learning-based studies, rely on the fact that the obtained genetic codes are easily human-understandable in terms of key features found, clusters size and classification algorithm setup, and summarize each attack class behavior. Furthermore, this approach provides a better versatility of the training phase that can be customized for each class of the classification problem, making the overall structure modular with respect to the problem-related classes: in fact, as new attacks need to be modelled, one can simply add the corresponding One-vs-All classifier to the ensemble, without need to retrain the whole system. This not only makes the training stage faster, but also makes the proposed approach appealing towards low-cost hardware.

7 CONCLUSIONS

In this work, we proposed a scalable and modular architecture to detect attacks in Wi-Fi networks by analyzing frames using a machine learning approach. The proposed system is made of an ensemble of One-Against-All classifiers each one targeting a specific attack class, suitably trained and optimized using a

Table 6: Performance comparison with related state-of-the-art works.

Related Works	**Dataset Size**	**Classification Problem**	**Method**	**Feature Selection Strategy**	**Performance Index**	**Performance (%)**
(Kolias et al., 2016)	Entire AWID dataset	Both using macro-classes and single attack labels	Random Tree J48	Manual	Accuracy	96.1982
(Aminanto and Kim, 2017)	AWID subset ($\sim$2 millions patterns)	4 macro-classes	Deep Learning	Automatic	Accuracy	92-98 (depending on class)
(Kolias et al., 2017)	Entire AWID dataset	Both using macro-classes and single attack labels	Ant Colony-based	Both manual and automatic	Accuracy	77.8
(Thing, 2017)	AWID subset ($\sim$2.4 millions patterns)	4 macro-classes	Deep Learning	Automatic	Accuracy	98.6688
(Qin et al., 2018)	AWID subset ($\sim$1.3 millions patterns)	4 macro-classes	SVM	Both manual and automatic	Accuracy	89-99 (depending on class)
(Sethuraman et al., 2019)	Modified subset of the AWID dataset	7 attack classes	HMM and KDE	Automatic	Precision	92.28
(Rizzi et al., 2020)	AWID subset ($\sim$84 millions patterns)	14 attack classes	BSAS and K-NN driven by GA	Both manual and automatic	Accuracy	92.446 (false alarm rate 12.1)
This work	AWID subset ($\sim$84 millions patterns)	14 attack classes	BSAS and K-NN driven by GA	Both manual and automatic	Accuracy	91.1 (false alarm rate 5.9)

genetic algorithm.

This approach let us simplify the classification problem by maximizing the training capabilities with a smaller training set, leading to a reduced amount of dissimilarity measure computations between patterns. During the training phase, we optimize the granulation strategy hyper-parameters (θ, in this case) for each attack class, creating optimized training subset of patterns. The very same optimization procedure performs features selection in order to simplify data processing by each classifier and to extract knowledge about recurrent characteristics of each attack. Correctness of knowledge discovery capabilities have been evaluated by comparing results with attack tools output (see Section 6). Furthermore, the proposed approach has been designed in order to have a natively parallel design, easily employable on multi-threaded architectures on general purpose multi-core systems, or on a dedicated hardware implementation on FPGA or ASIC circuits.

Intrusion detection is performed by analyzing network traffic frame-by-frame, obtaining performance values that show an average of more then 90% of frames correctly classified, despite of the difficulties to model the complex normal traffic process. With respect to our previous system performances (Rizzi et al., 2020), we have obtained comparable results in terms of accuracy, while greatly improving the false alarm rate (see Table 6). These results are comparable with state-of-the-art works whilst providing the further advantages explained before. To further improve the reliability of the classifier and the knowledge discovery capabilities, in particular for those attacks that do not allow single frame analysis with such a high success rate, higher complexity tools should be developed, that look through *sequences* of frames. In future works we will study the impact of strategies that let us represent Wi-Fi patterns in different mathematical spaces, able to enhance machine learning algorithms capabilities to classify and cluster patterns whilst maintaining white-box models that, if combined with automatic features selection methods, are able to output human readable filtering rules. Preliminary works show that a subset of attacks is recognizable more easily considering sequences of frames conveniently collected and ordered.

REFERENCES

Abhilash, G. and Divyansh, G. (2018). Intrusion detection and prevention in software defined networking. In *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–4.

Agarwal, M., Biswas, S., and Nandi, S. (2015). Detection of de-authentication dos attacks in wi-fi networks: A machine learning approach. In *2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 246–251.

Aminanto, M. E. and Kim, K. (2017). Detecting impersonation attack in wifi networks using deep learning approach. In Choi, D. and Guilley, S., editors, *Information Security Applications*, pages 136–147, Cham. Springer International Publishing.

Anton, S. D. D., Fraunholz, D., and Schotten, H. D. (2019). Using temporal and topological features for intrusion detection in operational networks. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, New York, NY, USA. Association for Computing Machinery.

Baldini, L., Martino, A., and Rizzi, A. (2019). Stochastic information granules extraction for graph embedding and classification. In *Proceedings of the 11th International Joint Conference on Computational Intelligence - Volume 1: NCTA, (IJCCI 2019)*, pages 391–402. INSTICC, SciTePress.

Benzaïd, C., Boulgheraif, A., Dahmane, F. Z., Al-Nemrat, A., and Zeraoulia, K. (2016). Intelligent Detection of MAC Spoofing Attack in 802.11 Network. In *Proceedings of the 17th International Conference on*

Distributed Computing and Networking, ICDCN '16, pages 47:1–47:5, New York, NY, USA. ACM.

Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2014). Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, 16(1):303–336.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

El-Khatib, K. (2010). Impact of Feature Reduction on the Efficiency of Wireless Intrusion Detection Systems. *IEEE Transactions on Parallel and Distributed Systems*, 21(8):1143–1149.

Frank, E., Hall, M. A., and Witten, I. H. (2016). *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann, 4 edition.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Guennoun, M., Lbekkouri, A., and El-Khatib, K. (2008). Selecting the Best Set of Features for Efficient Intrusion Detection in 802.11 Networks. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–4.

IEEE (2016). Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534.

Kolias, C., Kambourakis, G., Stavrou, A., and Gritzalis, S. (2016). Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Communications Surveys Tutorials*, 18(1):184–208.

Kolias, C., Kolias, V., and Kambourakis, G. (2017). Termid: A distributed swarm intelligence-based approach for wireless intrusion detection. *Int. J. Inf. Secur.*, 16(4):401–416.

Lanze, F., Panchenko, A., Braatz, B., and Engel, T. (2014). Letting the Puss in Boots Sweat: Detecting Fake Access Points Using Dependency of Clock Skews on Temperature. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 3–14.

Martino, A., Frattale Mascioli, F. M., and Rizzi, A. (2020a). On the optimization of embedding spaces via information granulation for pattern recognition. In *2020 International Joint Conference on Neural Networks (IJCNN)*. Accepted for Publication.

Martino, A., Giuliani, A., and Rizzi, A. (2019a). (hyper)graph embedding and classification via simplicial complexes. *Algorithms*, 12(11).

Martino, A., Giuliani, A., Todde, V., Bizzarri, M., and Rizzi, A. (2020b). Metabolic networks classification and knowledge discovery by information granulation. *Computational Biology and Chemistry*, 84:107187.

Martino, A., Rizzi, A., and Frattale Mascioli, F. M. (2017). Efficient approaches for solving the large-scale k-medoids problem. In *Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: IJCCI,*, pages 338–347. INSTICC, SciTePress.

Martino, A., Rizzi, A., and Frattale Mascioli, F. M. (2019b). Efficient approaches for solving the large-scale k-medoids problem: Towards structured data. In Sabourin, C., Merelo, J. J., Madani, K., and Warwick, K., editors, *Computational Intelligence: 9th International Joint Conference, IJCCI 2017 Funchal-Madeira, Portugal, November 1-3, 2017 Revised Selected Papers*, pages 199–219. Springer International Publishing, Cham.

Powers, D. M. W. (2011). Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63.

Qin, Y., Li, B., Yang, M., and Yan, Z. (2018). Attack detection for wireless enterprise network: a machine learning approach. In *2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–6.

Rizzi, A., Granato, G., and Baiocchi, A. (2020). Frame-by-frame wi-fi attack detection algorithm with scalable and modular machine-learning design. *Applied Soft Computing*, 91:106188.

Roux, J., Alata, E., Auriol, G., Kaâniche, M., Nicomette, V., and Cayre, R. (2018). Radiot: Radio communications intrusion detection for iot - a protocol independent approach. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–8.

Sethuraman, S. C., Dhamodaran, S., and Vijayakumar, V. (2019). Intrusion detection system for detecting wireless attacks in ieee 802.11 networks. *IET Networks*, 8(4):219–232.

Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., and Stiller, B. (2010). An Overview of IP Flow-Based Intrusion Detection. *IEEE Communications Surveys Tutorials*, 12(3):343–356.

Takahashi, D., Xiao, Y., Zhang, Y., Chatzimisios, P., and Chen, H.-H. (2010). IEEE 802.11 User Fingerprinting and Its Applications for Intrusion Detection. *Comput. Math. Appl.*, 60(2):307–318.

Theodoridis, S. and Koutroumbas, K. (2008). *Pattern Recognition*. Academic Press, 4 edition.

Thing, V. L. L. (2017). IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6.

Youden, W. J. (1950). Index for rating diagnostic tests. *Cancer*, 3(1):32–35.