A Data Fusion Technique to Detect Wireless Network Virtual Jamming Attacks

G. Escudero-Andreu, K.G. Kyriakopoulos, F.J. Aparicio-Navarro and D.J. Parish
School of Electronic, Electrical and Systems Engineering
Loughborough University
LE11 3TU Loughborough, U.K.
{elge2, elkk, elfja2, d.j.parish}@lboro.ac.uk

D. Santoro, M. Vadursi
Department of Engineering
University of Naples "Parthenope"
Naples, Italy
{diego.santoro; michele.vadursi}@uniparthenope.it

***Abstract*— Wireless communications are potentially exposed to jamming due to the openness of the medium and, in particular, to virtual jamming, which allows more energy-efficient attacks. In this paper we tackle the problem of virtual jamming attacks on IEEE 802.11 networks and present a data fusion solution for the detection of a type of virtual jamming attack (namely, NAV attacks), based on the real-time monitoring of a set of metrics. The detection performance is evaluated in a number of real scenarios.**

***Index Terms*—Measurements and networking, measurements on wireless networks, virtual jamming attacks.**

I. Introduction

The easily accessible shared transmission channel of wireless systems is both an advantage in terms of usability but also poses an important vulnerability aspect. The open nature of wireless networks exposes the communication channel to a number of attacks that are difficult to trace [1], [2], such as jamming. The jamming attack is one of the most serious threats to wireless systems. Today, jamming attacks are very common and rather easy to implement, considering the number of off-the-shelf tools available ([3], [4] and [5]). Jamming attacks can be classified into physical and virtual jamming. Examples of the former are: radio jamming, where the attacker transmits continuously a radio signal carrying random bits, and the collision attack, where the attacker sends a packet only when it senses that a legitimate user is sending a valid packet in order to cause a collision [6]. Examples of virtual jamming are the spurious RTS/CTS attack, which is carried out by sending fake RTS frames, and NAV attacks, where the attacker alters the duration field of legitimate packets. Both attacks aim to delay legal frame transmissions. Unlike physical jamming, virtual jamming is very easy to implement and needs little power to carry out the attack.

The literature provides solutions, which deal with a wide range of jamming threats in simulated environments like Network Simulation (NS) [7] and GloMoSim [8] or introduces MAC layer changes in the IEEE 802.11 protocol. In this work, we tackle the problem of the virtual jamming attacks and propose a new solution to detect NAV attacks based on Dempster-Shafer (DS) theory. Data fusion has been widely used for traffic classification purposes [9], while the proposed solution has been previously proven to successfully detect de-authentication attack and Man-in-the-Middle attacks on 802.11 networks [10].

The rest of this paper is organised as follows. Section II analyses the state of the art for jamming attacks in IEEE 802.11 networks. An explanation of our attack is described in Section III, introducing a novel mechanism to prevent the attack along with the best metrics to detect it. In Section IV, the implementation of the attack and the testbed are described in detail, concluding with the analysis of the detection results. Finally, in Section V the conclusions are presented.

II. Related Work

Jamming attacks have been widely researched in the past [11] – [16], offering multiple solutions which tackle the problem for a wide range of jamming attacks.

In [11] the authors present DOMINO, a piece of software installed in or near an access point in order to detect MAC Layer greedy behaviour in 802.11 hotspots. DOMINO is organised in three modules: (i) Deviation Estimation Component (DEC), (ii) Anomaly Detection Component (ADC) and (iii) Decision Making Component (DMC). The DEC module performs the following tests: retransmission consistency, DIFS consistency, NAV consistency and backoff manipulation test. DOMINO runs the tests for each node by tracking each node's transmissions in the network. Therefore, the way the tests are computed might become time and resource consuming when the number of nodes is increased [12]. DOMINO's performance is assessed through the simulator NS. The results show that DOMINO is characterised by high accuracy of detection and resiliency to several factors, such as traffic type.

In [13], the authors propose a distributed cross-layer detection system for a wide range of jamming attacks. The monitoring functionality is randomly distributed among the nodes and the detection mechanism is organised in two phases. In the first phase, the system performs 4 tests on: (i) the physical idle time, (ii) the average number of RTS/CTS frames transmitted by a node, (iii) the virtual idle time (NAV) and (iv) the average number of retransmissions of a node. In the second phase, the results are put all together and then a final test is carried out in order to increase the accuracy. It's worth noting

that the solution presented in [13] needs to carry out the tests for each node in the network. Therefore, the solution is time and resource consuming when increasing the number of the nodes. The performance is assessed through the simulator GloMoSim and the results show that: (i) as the number of nodes increases, the more the DRD (Data Rate Detection) decreases, and (ii) the number of false positive increases when increasing the number of the nodes.

In [14] the authors propose a detection algorithm based on thresholds that is able to detect and classify physical and virtual jamming attacks. The algorithm needs the following metrics as inputs: PDR (Packet Delivery Ratio) and PSR (Packets Send Ratio). The algorithm's outcomes are compared with a Signal Strength Consistency check in order to improve the overall system's accuracy. The consistency test is necessary because, as the authors suggest, the PDR might be low since a node might be running out of battery or if user is moving out of the coverage area. The used metrics have to be calculated for each node, and moreover data has to be retrieved from transmitting and receiving nodes during the jamming attack. The way data are collected during the attack makes the solution an off-line solution. The authors provide simulated results by using NS which are characterised by high accuracy and precision rates.

In [15] the authors propose a distributed solution to detect jamming attacks at the physical layer. In detail, the method is based on the detection of changes in the statistical characteristics of the Signal to Noise Ratio (SNR). The detection is carried out locally by using either a simple-threshold algorithm or a cusum-type algorithm. The authors also present an improved version based on DS theory characterised by high Detection Rates (DRs) and low False Positive Rates (FPrs). The improved version based on cusum algorithm provides very high DRs and FPrs while the improved version based on the simple threshold algorithm provides a further increase of 80% of the DRs and FPrs. The authors do not provide any assessment about algorithm performance in scenarios like hidden terminal or fading channel.

Finally, in [16] the authors focus on virtual jamming attacks, providing a protection scheme, which is implemented in the MAC layer. In detail, it needs two MAC layer timers as well as a NAV timer. The RTS timer and the CTS timer, to track the RTS-DATA and the CTS-ACK sequences. Regarding the RTS timer, if no header data is received at the beginning of the RTS time, then the NAV time is set to zero and the node is ready to transmit. Regarding the CTS timer, if no expected ACK frame is received after the CTS frame, the NAV timer will be set to zero and the node is ready to transmit. The overall solution represents a variant from the original IEEE 802.11 protocol. The performances were assessed by using NS. The results show that during a virtual jamming attack the throughput of legitimate users do not experience any effects.

In this work, we present a new solution to detect NAV attack on 802.11 networks. Unlike the solutions presented earlier, it is implemented as a light, centralised and on-line solution. The detector is implemented as a single monitoring machine, which derives the metrics by observing the ongoing traffic. The performance of our solution has been assessed in several real scenarios providing good results in terms of high DRs, low FPrs and FNrs (False Negative Rates).

III. Proposed Method

A. Attack Description

The NAV attack exploits the virtual-sense mechanism, which aims to mitigate the collisions resulting from the hidden-terminal problem. Specifically, the header of each 802.11 packets contains a particular field, named duration, which determines the time (in milliseconds) needed to transmit the packet on the channel and the time interval during which the channel will be busy. Every network node reads the value of the duration field in order to set its own NAV timer. Assuming that the channel is busy and another node has something to transmit, those nodes will have to wait a period equal to NAV in order to transmit. After that, those nodes start decreasing their backoff time again. Finally, when the backoff timer reaches zero, if the channel is idle then the nodes start transmitting otherwise, they defer their transmission once again.

To carry out a NAV attack, the attacker needs to overwrite two mechanisms of the IEEE 802.11 protocol: the RTS/CTS mechanism and the procedure to calculate the backoff time. With regards to the RTS/CTS mechanism, the field duration of each RTS packet is set to the maximum NAV value 32767 (32ms). All nodes listening on the channel will set their NAV timers. On the other hand, the contention window of the backoff calculation mechanism is set to zero so that the attacker is able to transmit in the very first free time slot.

B. Attack Detection Methodology

The proposed attack detection methodology is based on the use of evidence theory. In recent years, the theory of belief functions, also known as the evidence theory developed by Dempster and Shafer [17], has drawn the attention of many researchers, especially in the fields of sensor and data fusion [18]. DS theory provides a simple framework to merge information coming from different sensors (DS rule), taking into account all the available evidence in uncertain situations. The DS theory does not require prior knowledge, enables a way of measuring ignorance, when the evaluated data cannot be allocated within the normal or abnormal hypothesis. Furthermore, it has also proved to be a viable solution in cases where it is impossible to apply classical sensor fusion techniques, such as Kalman filter or Bayesian networks, or even when it is virtually impossible to find a pattern in the system behaviour to build an appropriate model [15]. Although DS theory inflicts additional computational load when computing mass functions, reducing the number of hypothesis to three makes this theory applicable to a live detection system.

Before introducing our solution in detail, we briefly introduce some basic concepts about DS theory.

1) Dempster-Shafer Theory

The DS theory starts by taking into account a set of events $\Theta = \{\theta_1, \theta_2, \dots, \theta_n\}$ (known as Frame of Discernment), which is a finite set of all possible mutually exclusive propositions and hypotheses about some problem domain. The total number

of subsets of Θ, defined by the number of hypotheses, is defined by the power set $P(\Theta)$.

Each subset from Θ is assigned a probability value, or a confidence interval within [0, 1], by an observer from the mass probability function m using the basic probability assignment function. The Basic Probability Assignment function (BPA) is defined as a function m: $P(\Theta) \rightarrow [0, 1]$ satisfying the following three conditions [17]:

$$m(\emptyset) = 0 \quad (1)$$

$$m(A) \geq 0, \forall\, A \subseteq \Theta \quad (2)$$

$$\sum_{A \subseteq \Theta} m(A) = 1 \quad (3)$$

The function $m(A)$ describes the measure of belief that is committed exactly to the hypothesis A. It is worth noting that, in contrast to probability theory, DS theory does not comply with the Additivity Rule.

The DS theory is a technique that combines evidence of information from different observers with similar Θ using the DS rule of combination [17]. Let $m_1(A)$ and $m_2(A)$ be the BPA in the hypothesis A, from observer 1 and 2, respectively. DS rule of combination calculates the orthogonal summation of the BPAs values in one hypothesis from two different observers into a single belief ($m_1(A) \oplus m_2(A)$). After defining the BPA value for each hypothesis, the information from the sensors is merged as follows:

$$m_{comb}(H) = \frac{\sum_{X \cap Y = H} m_1(X) * m_2(Y)}{1 - \sum_{X \cap Y = \emptyset} m_1(X) * m_2(Y)} \; \forall\, H \neq \emptyset \quad (4)$$

Further details on DS theory can be found in [14].

2) Proposed Detection Method

A hard and delicate issue that affects the developing phase of a detector is how to define the BPA values. In the literature there exist multiple ways of assigning probabilities to each of the hypotheses in DS theory, ranging from data mining techniques to empirical approaches. However, none of these works provide methods to find an automatic and self-adaptive process of BPA, and few of them could be used off-the-shelf without a previous training or fine tuning period.

The proposed methodology to detect virtual jamming is based on the basic probability functions proposed in [10]. The framework proved to be successful detecting one type of Man-in-the-Middle attack and de-authentication attack on wireless networks. Additionally, the proposed solution also proved to be fast enough to detect the attacks in real-time.

Initially, the algorithm gathers a number of incoming frames and, for each frame, a series of metrics are extracted. Then, using the gathered frames, the statistical characteristics of each individual metric are calculated. In particular, the mean and the mode that are used as reference of normality, as well as the distribution of the data. These statistical parameters represent the signature of the well-behaved wireless clients, and these are compared to the metric values of each individual frame that is analysed. After computing all these statistical parameters, the BPA value for each individual metric is calculated using the methodology proposed in [10].

For our experiments, the three different hypotheses are *{A} Attack* (the analysed frame is malicious), *{N} Normal* (the analysed frame is non-malicious) and *{U} Uncertainty* (refers to the level of ignorance).

After the different BPAs are calculated, these values are fused using the DS theory. The DS rule allows combining evidence from only two observers at a time. Therefore, the different BPA values are combined after several consecutive iterations. The resulting BPAs indicate which hypotheses receive the highest evidence of support.

It is worth noting that the proposed methodology works in a sliding window scheme, as a FIFO queue of prefixed size. If the currently analysed frame is judged to be normal, it is pushed to the end of the sliding window, the oldest frame is pushed out of the window, and the analysed frame will be used to compute new BPA values. Otherwise, the current frame is simply dropped. The window size affects the algorithm performance and side effects are investigated in Section IV.

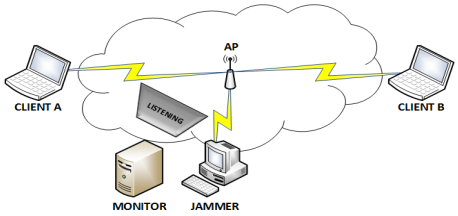

Fig. 1. Testbed architecture

IV. Performance Assessment

A. Testbed Description

To assess the detection method performance, an experimental testbed has been set up and it is depicted in Fig. 2. It comprises 3 types of nodes, as well as an Access Point (AP):

1. Attacker. It runs on Linux Ubuntu 10.04 Lucid Lynx. The NIC (Network Interface Controller) is equipped with an Atheros 5100 chip, which is supported by the legacy driver ATH5000 for Atheros chipsets. The RTS/CTS mechanism and backoff calculation mechanism have been modified ad hoc, as described in Section III. The driver is loaded as a new module in the kernel, forcing an automated bound with the hardware during the system initialisation.
2. Monitor. This station is a monitor node equipped with the same Atheros chipset as the attacker and it is configured in Monitor Mode to listen to the channel. We used Wireshark [19] to collect the traffic. Furthermore, we used our modified version of ATH5000 driver to gather live statistics from the wireless interface card, i.e. CRC error. The monitor node is placed close the AP.
3. Client. We used two clients during our experiments, namely $Client_A$ and $Client_B$. Both clients act as well-behaved users, sending traffic during the whole monitoring period. The traffic was artificially generated using the iperf linux command [20] to send constant bit rate UDP traffic in data frames.

The experimental campaign is summarised in Table I. In detail, in order to validate our solution, a list of 9 scenarios is proposed, including fixed and mobile well-behaved users. The

first two scenarios are normal scenarios without any attack. As regards scenarios from 3 to 9, each of them has a total duration of 90 seconds and includes three phases: i) initial phase, where only the well-behaved node/s send traffic, ii) attack phase, where the attack is launched, and iii) final phase, where the attacker is deactivated in order to recover normality. Each phase has a duration of 30 seconds.

TABLE I. EXPERIMENT DESCRIPTION

Scenario #	Description
1	No attacker, fixed ClientA and ClientB
2	No attacker, fixed ClientA and fixed ClientB (with high NAV value)
3	Fixed ClientA
4	Fixed ClientA and ClientB
5	Moving ClientA
6	Moving ClientA and ClientB
7	Fixed ClientA and moving ClientB
8	Moving ClientA and fixed ClientB
9	Fixed ClientB sending TCP traffic

The test cases have been designed to emulate actual scenarios in WiFi networks. They include scenarios where a client is located in a fixed position, keeping a constant distance to the AP to maintain stable parameters in the received radio signal, as well as situations in which random movement was introduced to emulate an actual mobile situation. The movement reproduced a normal walking pace within an indoor environment, keeping the distance between 1 meter, equivalent to the fixed position, up to a maximum of 10 meters far from the AP. Mixed scenarios are also taken into account, including both fixed and moving nodes to assess the performance of the detection algorithm within the same room, when multiple clients are competing for the available resources. To conclude, a test using TCP was carried out to study how the jamming attack could affect the establishment of a TCP connection, which requires the completion of a three-way handshake process successfully to enable a communication link.

B. Metrics

Regarding the metrics to monitor, the presented literature drove us through the selection process. The metrics monitored are: NAV, frame deltatime and FSN (Frame Sequence Number). Monitoring NAV to detect a greedy behaviour or NAV attack is common in the literature [11], [13] and [16]. However, detecting intelligent jamming attacks by simply monitoring NAV is not a robust solution, because even if the intelligent jammer exploits high values of the NAV to carry out the attack, those values remain legitimate [21]. In [13], [14] and [21], the authors suggest using metrics related to the number of packets sent and received such as PDR, PSR and channel utilisation, in order to detect several types of attacks. We included in our analysis a similar metric: frames deltatime. Frame deltatime is obviously affected by the attack since the main effect of a jamming attack is a service disruption, causing bandwidth reduction and an increase in the deltatime between two consecutive transmitted frames. Finally, we took into consideration the FSN metric which has detectable peaks in the first order time differences of the frame sequence numbers, DiffFSN. The FSN metric presents the aforementioned peaks in its first order time difference because the WiFi card buffers overflow during the attack and it causes dropped packets.

Finally, the performance of our solution is evaluated by using the following metrics: (i) Detection Rate $DR = \frac{TP}{TP+FN}$ where TP stands for True Positive and FN stands for False Negative, (ii) False Positive rate $FPr = \frac{FP}{FP+FN}$ where FP stands for False Positive and FN stands for False Negative and (iii) False Negative rate $FNr = \frac{FN}{TP+FN}$.

C. Experimental Results

The presented solution has been preliminary tested on the first two scenarios in Table I, when no attack is ongoing, in order to check the performance of our solution in term of FPr.

The scenario 1 has two legitimate nodes with both low NAV value, whereas the scenario 2 has two legitimate nodes but one with a low NAV value and the other one with a naturally high NAV value.

The experimental results for these two scenarios are reported in Table II, for each combination of metrics. The results show that the best single metric combination in term of lowest FPrs is the NAV. However, while the FPr for the NAV metric is around 2% in scenario 1, it rises up to 17% in scenario 2. Such worsening in the performance is due to the fact that in scenario 2, one of the legitimate users has set a high NAV value, which confuses the algorithm and causes false positive. In fact, the frames with the high NAV value are 17% of the total frames in the second scenario and all of them have been detected as malicious frames. As regards the multiple metric combinations, all of them exhibit similar performance, with an FPr equal to (about) 25%, which unfortunately does not represent a performance enhancement in scenario 2. The detection algorithm thus suffers from false positives when legitimate users have high NAV values.

TABLE II. FPRS FOR SCENARIOS 1 AND 2

	Scenario 1	Scenario 2
NAV	2%	19%
DiffFSN	25%	38%
Deltatime	86%	86%
NAV, Deltatime	86%	86%
NAV, DiffFSN	25%	25%
Deltatime, DiffFSN	23%	25%
Deltatime, DiffFSN, NAV	23%	25%

The experimental results are presented in Table III. The table reports the performance evaluation of our methodology in terms of DRrs, FPrs and FNrs for different combination of metrics and for each scenario in Table I. The results presented in Table II have been obtained using a sliding window size of 20 samples.

Among all the evaluated combination of metrics, making use of the single metric NAV provides the best results in terms

of high DRs, low FNrs and FPrs in all the tested scenarios. Fig.3 shows the NAV metric over time.

The single metric Deltatime exhibits high DRs with likewise high FPrs. These bad results are due to the fact that legitimate clients cannot send any frames over the wireless medium during the attack and, in consequence, the monitoring system cannot update the metrics for the calculation of the respective beliefs. Fig. 4 shows the Deltatime over time. The evidence of an attack is visible in the Deltatime only at end of the attack apart from some occasional spikes during the attack.

The single metric DiffFSN exhibits the worst results among the single metric combinations with low DRs, low FPr and high FNr. The DiffFSN metric, like the metric Deltatime, cannot be update by the monitoring system during the attack. Fig. 5 shows the DiffFSN over time. The evidence of an attack is visible in the DiffFSN only at end of the attack.

The multiple metrics combinations do not provide any improvement in comparison with the result obtained by the single metric combination NAV.

In detail, the multiple metrics combination NAV, Deltatime provides high DR but high FPr. It worth noting that the results are very similar to the single metric combination Deltatime. No improvements are shown bringing in the further evidence provided by Deltatime metric.

Finally, the multiple metrics combinations NAV, DiffFSN and NAV, Deltatime and DiffFSN provide the worst results with low DRs, high FNrs and low FPrs.

The best metrics combination in term of DRs, FNrs and FPrs is the single metric combination NAV. A graphical summary of the results discussion is provided in fig. 5, which depicts bars plots of the minimum, average and maximum value for the considered performances indicators grouped by metric combination among all the scenarios. However, any solution to detect NAV attack based on the single metric NAV is completely lack of robustness as the NAV value takes totally legitimate values during the attack.

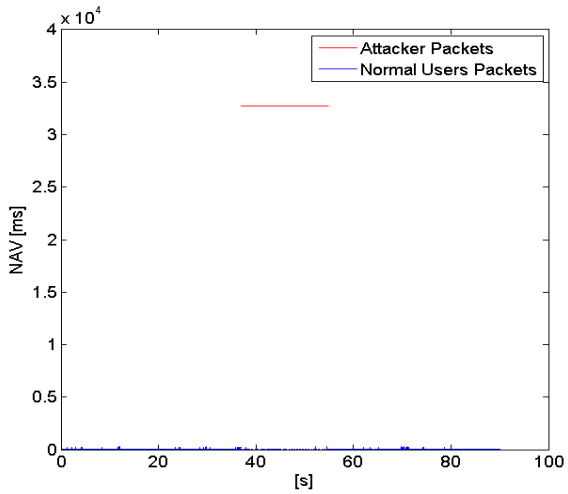

Fig. 2. NAV metric in Scenario 7.

V. Conclusions

In this work, we have tackled the problem of identifying virtual jamming attacks on IEEE 802.11 networks and presented a solution based on DS theory aimed at detecting NAV attacks.

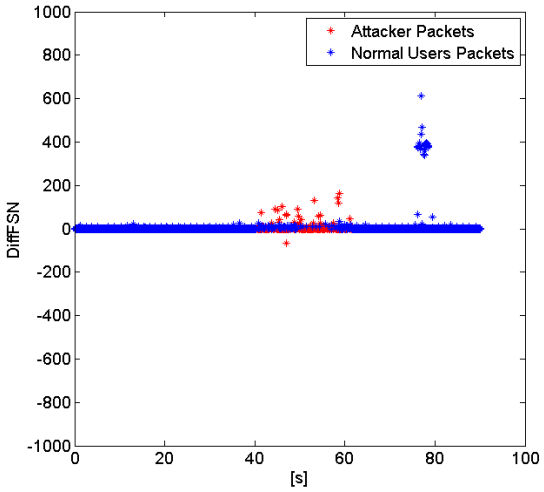

Fig. 3. DiffFSN in Scenario 6.

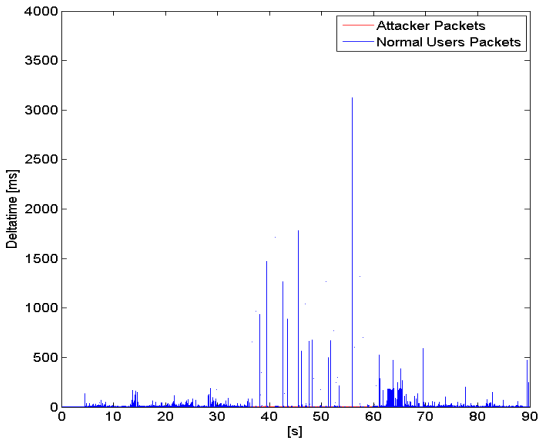

Fig. 4. Deltatime in Scenario 4.

The solution has been tested on real wireless scenarios, showing good results in terms of DRs and FNrs. However, the proposed detection method suffers from high FPr when there is no attack ongoing and one of the users has legitimately high NAV values. Future work will be focused on the mitigation of the effects of FPr, which prevent the proposed methodology from being considered as a viable solution in operating networks. In particular, ongoing activity aims to investigate further metrics to be used in combination with NAV, in order to have a more robust solution.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) Grant number EP/K014307/1 and the MOD University Research Collaboration in Signal Processing.

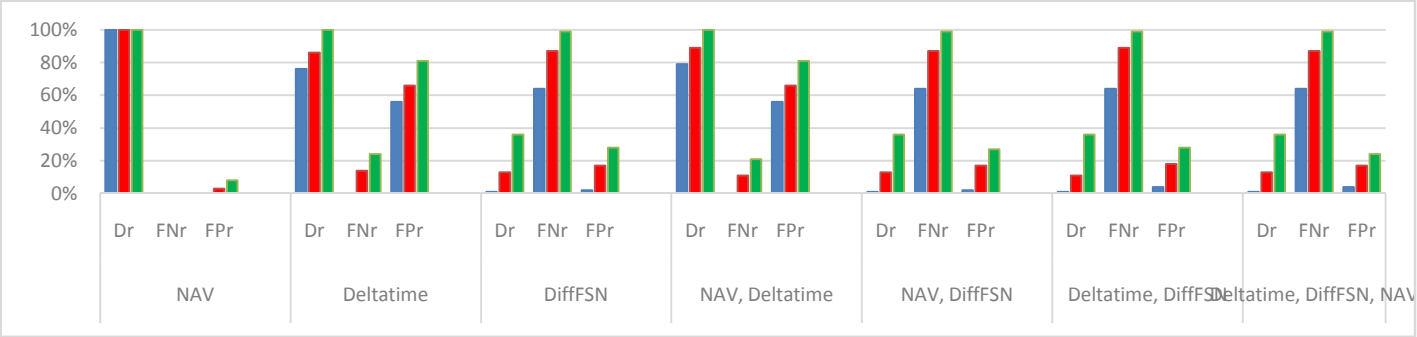

Fig. 5. Results in bars plots. Blue-painted bars, red-painted bars and green-painted bars represents respectively the minimum, the average and the maximum value of the considered performance indicator among all the scenarios for the different combination metrics.

TABLE III. RESULTS

	Scenario 3			Scenario 4			Scenario 5			Scenario 6		
	Dr	**FNr**	**FPr**	**Dr**	**FNr**	**FPr**	**Dr**	**FNr**	**FPr**	**Dr**	**FNr**	**FPr**
NAV	100%	0%	0%	100%	0%	8%	100%	0%	0%	100%	0%	3%
Deltatime	83%	17%	60%	99%	1%	64%	83%	17%	56%	79%	21%	69%
DiffFSN	1%	99%	28%	2%	98%	7%	1%	99%	27%	30%	70%	20%
NAV, Deltatime	83%	17%	60%	99%	1%	64%	83%	17%	56%	79%	21%	69%
NAV, DiffFSN	1%	99%	24%	3%	97%	7%	1%	99%	27%	30%	70%	20%
Deltatime, DiffFSN	1%	99%	28%	3%	97%	17%	1%	99%	23%	23%	77%	18%
Deltatime, DiffFSN, NAV	1%	99%	24%	15%	85%	17%	1%	99%	23%	23%	77%	18%
	Scenario 7			**Scenario 8**			**Scenario 9**					
NAV	100%	0%	3%	100%	0%	2%	100%	0%	6%			
Deltatime	81%	19%	69%	76%	24%	64%	100%	0%	81%			
DiffFSN	17%	83%	20%	36%	64%	17%	1%	99%	2%			
NAV, Deltatime	81%	19%	69%	98%	2%	64%	100%	0%	81%			
NAV, DiffFSN	17%	83%	20%	36%	64%	17%	1%	99%	2%			
Deltatime, DiffFSN	14%	86%	19%	36%	64%	16%	1%	99%	4%			
Deltatime, DiffFSN, NAV	14%	86%	19%	36%	64%	16%	2%	98%	4%			

REFERENCES

[1] N. Nostro, A. Ceccarelli, A. Bondavalli, F. Brancati, "A methodology and supporting techniques for the quantitative assessment of insider threats," *Proc. of the 2nd Int. Workshop on Dependability Issues in Cloud Computing* (p. 3), 2013, ACM.

[2] N. Nostro, A. Ceccarelli, A. Bondavalli, and F. Brancati. "Insider Threat Assessment: a Model-Based Methodology," *SIGOPS Oper. Syst. Rev.*, 48, 2, Dec. 2014, pp. 3-12.

[3] Sesp jammers, www.sesp.com (Access Date: 19 Jun, 2015)

[4] Mobiledevice jammer, www.phonejammer.com (Accessed 19 Jun'15)

[5] Software-defined radios, www.ettus.com (Accessed 19 Jun'15)

[6] A. Mahanti, N. Carlsson, C. Williamson, M. Arlitt, "Ambient Interference Effects in Wi-Fi Networks," *Proc. of the 9th IFIP TC 6 int. conf. on Networking,* vol. 6091, pp. 160-173, 2010.

[7] NS, www.nsnam.org (Accessed 19 Jun'15)

[8] X. Zeng, R. Bagrodia, M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," *Proc. of the Parallel and Distributed Simulation PADS 1998*, May 1998, pp. 154 – 161.

[9] A. Dainotti, A. Pescape', C. Sansone, "Early Classification of Network Traffic through Multi-classification" *Proc. of the 3rd Intern. Workshop on Traffic Monitoring and Analysis TMA 2011,* Lecture Notes in Computer Science, vol. 6613, pp. 122-135, April 2011.

[10] F.J. Aparicio-Navarro, K.G. Kyriakopoulos, D.J. Parish, "A multi-layer data fusion system for Wi-Fi attack detection using automatic belief assignment," *Proc. of World Congress on Internet Security,* Canada, pp. 45 – 50, Jun 2012.

[11] M. Raya, I. Aad, J.-P. Hubaux, A. El Fawal, "DOMINO: Detecting MAC Layer Greedy Behavior in IEEE 802.11 Hotspots," *IEEE Trans. Mobile Comput.*, Vol. 5, Issue 12, pp. 1691 – 1705, Dec. 2006.

[12] L. Montecchi, N. Nostro, A. Ceccarelli, G. Vella, A. Caruso, A. Bondavalli, "Model-based Evaluation of Scalability and Security Tradeoffs: a Case Study on a Multi-Service Platform," *Electr. Notes Theor. Comput. Sci*. 310: 113-133 (2015)

[13] G. Thamilarasu, S. Mishra, R. Sridhar, "A Cross-layer Approach to Detect Jamming Attacks in Wireless Ad hoc Networks," *Proc. of Military Comm. Conf. MILCOM 2006,* Oct. 2006.

[14] Le Wang, A.M. Wyglinski, "A combined approach for distinguishing different types of jamming attacks against wireless networks," *Proc. of Comm., Comp. and Sig. Proc. Conf.*, pp. 809-814, Aug. 2011.

[15] A.G. Fragkiadakis, V. A. Siris, N. E. Petroulakis, A. P. Traganitis, "Anomaly-based intrusion detection of jamming attacks, local versus collaborative detection," *Wirel. Comm. and Mob. Comp.*, 15 (2), pp. 276 – 294, Feb. 2015.

[16] D. Chen, J. Deng, and P. K. Varshney, "Protecting Wireless Networks against a Denial of Service Attack Based on Virtual Jamming," *ACM MobiCom '03,* San Diego, USA, Sept. 2003.

[17] G. Shafer, A Mathematical Theory of Evidence, Princeton Univ Press, 1976.

[18] D. Yu, D. Frincke, "Alert confidence fusion in intrusion detection systems with extended Dempster-Shafer theory, " *Proc. ACM 2005*, Kennesaw, USA, pp. 142 – 147, Mar. 2005.

[19] G, Combs, Wireshark-network protocol analyzer, www.wireshark.org

[20] Iperf, Website Available: www.iperf.fr (Accessed 19 Jun'15).

[21] A.Y. Dak, N.E.A. Khalid, S. Yahya, "A novel framework for jamming detection and classification in wireless networks, " *Proc. of Computing and Networking Tech. ICCNT 2012*, Aug. 2012, pp. 240 – 246.