Dear Author,

Here are the proofs of your article.

- You can submit your corrections **online**, via **e-mail** or by **fax**.
- For **online** submission please insert your corrections in the online correction form. Always indicate the line number to which the correction refers.
- You can also insert your corrections in the proof PDF and **email** the annotated PDF.
- For fax submission, please ensure that your corrections are clearly legible. Use a fine black pen and write the correction in the margin, not too close to the edge of the page.
- Remember to note the **journal title**, **article number**, and **your name** when sending your response via e-mail or fax.
- **Check** the metadata sheet to make sure that the header information, especially author names and the corresponding affiliations are correctly shown.
- **Check** the questions that may have arisen during copy editing and insert your answers/ corrections.
- **Check** that the text is complete and that all figures, tables and their legends are included. Also check the accuracy of special characters, equations, and electronic supplementary material if applicable. If necessary refer to the *Edited manuscript*.
- The publication of inaccurate data such as dosages and units can have serious consequences. Please take particular care that all such details are correct.
- Please **do not** make changes that involve only matters of style. We have generally introduced forms that follow the journal's style.
  Substantial changes in content, e.g., new results, corrected values, title and authorship are not allowed without the approval of the responsible editor. In such a case, please contact the Editorial Office and return his/her consent together with the proof.
- If we do not receive your corrections **within 48 hours**, we will send you a reminder.
- Your article will be published **Online First** approximately one week after receipt of your corrected proofs. This is the **official first publication** citable with the DOI. **Further changes are, therefore, not possible.**
- The **printed version** will follow in a forthcoming issue.

Please note

After online publication, subscribers (personal/institutional) to this journal will have access to the complete article via the DOI using the URL: http://dx.doi.org/[DOI].
If you would like to know when your article has been published online, take advantage of our free alert service. For registration and further information go to: http://www.springerlink.com.

Due to the electronic nature of the procedure, the manuscript and the original figures will only be returned to you on special request. When you return your corrections, please inform us if you would like to have these documents returned.

Metadata of the article that will be visualized in OnlineFirst

Please note: Images will appear in color online but will be printed in black and white.

ArticleTitle	Identification of probe request attacks in WLANs using neural networks	
Article Sub-Title		
Article CopyRight	Springer-Verlag London (This will be the copyright line in the final PDF)	
Journal Name	Neural Computing and Applications	

Corresponding Author	Family Name	**Ratnayake**
	Particle	
	Given Name	**Deepthi N.**
	Suffix	
	Division	Intelligent Systems Research Centre, Faculty of Computing
	Organization	London Metropolitan University
	Address	166-220 Holloway Road, N7 8DB, London, UK
	Email	d.ratnayake@londonmet.ac.uk
Author	Family Name	**Kazemian**
	Particle	
	Given Name	**Hassan B.**
	Suffix	
	Division	Intelligent Systems Research Centre, Faculty of Computing
	Organization	London Metropolitan University
	Address	166-220 Holloway Road, N7 8DB, London, UK
	Email	h.kazemian@londonmet.ac.uk
Author	Family Name	**Yusuf**
	Particle	
	Given Name	**Syed A.**
	Suffix	
	Division	Institute of Industrial Research
	Organization	University of Portsmouth
	Address	PO1 2UP, Portsmouth, UK
	Email	adnan.yusuf@port.ac.uk

	Received	13 February 2013
Schedule	Revised	
	Accepted	22 August 2013

Abstract

Any sniffer can see the information sent through unprotected ‘probe request messages’ and ‘probe response messages’ in wireless local area networks (WLAN). A station (STA) can send probe requests to trigger probe responses by simply spoofing a genuine media access control (MAC) address to deceive access point (AP) controlled access list. Adversaries exploit these weaknesses to flood APs with probe requests, which can generate a denial of service (DoS) to genuine STAs. The research examines traffic of a WLAN using supervised feed-forward neural network classifier to identify genuine frames from rogue frames. The novel feature of this approach is to capture the genuine user and attacker training data separately and label them prior to training without network administrator’s intervention. The model’s performance is validated using self-consistency and fivefold cross-validation tests. The simulation is comprehensive and takes into account the real-world environment. The results show that this approach detects probe request attacks extremely well.

This solution also detects an attack during an early stage of the communication, so that it can prevent any other attacks when an adversary contemplates to start breaking into the network.

Keywords (separated by '-')	Wireless LAN - Intrusion detection - Real-time systems - IEEE 802.11 - DoS attacks - Feed-forward neural networks
Footnote Information	

Neural Comput & Applic
DOI 10.1007/s00521-013-1478-8

ORIGINAL ARTICLE

Identification of probe request attacks in WLANs using neural networks

Deepthi N. Ratnayake · Hassan B. Kazemian · Syed A. Yusuf

Received: 13 February 2013 / Accepted: 22 August 2013

Abstract Any sniffer can see the information sent through unprotected 'probe request messages' and 'probe response messages' in wireless local area networks (WLAN). A station (STA) can send probe requests to trigger probe responses by simply spoofing a genuine media access control (MAC) address to deceive access point (AP) controlled access list. Adversaries exploit these weaknesses to flood APs with probe requests, which can generate a denial of service (DoS) to genuine STAs. The research examines traffic of a WLAN using supervised feed-forward neural network classifier to identify genuine frames from rogue frames. The novel feature of this approach is to capture the genuine user and attacker training data separately and label them prior to training without network administrator's intervention. The model's performance is validated using self-consistency and five-fold cross-validation tests. The simulation is comprehensive and takes into account the real-world environment. The results show that this approach detects probe request attacks extremely well. This solution also detects an attack during an early stage of the communication, so that it can prevent any other attacks when an adversary contemplates to start breaking into the network.

D. N. Ratnayake (✉) · H. B. Kazemian
Intelligent Systems Research Centre, Faculty of Computing, London Metropolitan University, 166-220 Holloway Road, London N7 8DB, UK
e-mail: d.ratnayake@londonmet.ac.uk

H. B. Kazemian
e-mail: h.kazemian@londonmet.ac.uk

S. A. Yusuf
Institute of Industrial Research, University of Portsmouth, Portsmouth PO1 2UP, UK
e-mail: adnan.yusuf@port.ac.uk

Keywords Wireless LAN · Intrusion detection · Real-time systems · IEEE 802.11 · DoS attacks · Feed-forward neural networks

1 Introduction

Institute of Electrical and Electronic Engineers (IEEE) Wireless Local Area Networks (WLAN) are based on IEEE 802.11 protocol. The reliability of the media access control (MAC) layer of the IEEE 802.11 protocol is maintained by enforcing a response message from the access point (AP) for every request message from a station (STA). Attackers exploit this request-and-respond design flaw to generate probe request flood (PRF) attacks. Flooding attacks create severe performance dilapidations or decline of resources to genuine STAs when besieged by requests. Unprotected beacon or probe request and probe response frames which are sent 'clear' increase the risk of susceptibility. Usually, probing is the initial phase of any other attack in computer networks [1–7].

Evaluations of detection systems require identification of genuine and rogue frames in the sample. Analysing frames of a WLAN test bed manually or statistically and detecting a rogue frame are possible to some extent due to its controlled nature. However, identifying rogue frames from genuine frames in a real network completely is an impossible task. Therefore, researchers use existing sample datasets or other sanitised or simulated traffic to develop and test intrusion detection proposals. Although they are rich in variety of genuine and attack traffic, and considered as a benchmark for evaluating security detection mechanisms, these datasets do not contain background noise that a real-world dataset consists of. Therefore, the solutions that develop based on these datasets may not work as

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | ☐ LE | ☐ TYPESET
MS Code : NCA-3163 | ☑ CP | ☑ DISK

efficiently and effectively as they claim to be in real-world environments. This research investigates and analyses the traffic of a real-world WLAN and, therefore, works with actual WLAN frames [8–10].

The research seeks to identify existence of an adversary in a WLAN at the beginning of a frame transmission, so that it can prevent more disruptive attacks an adversary may plan to perform. The research learnt that traffic patterns are unforeseeable and have inherent complexities due to many factors including usage, the operating system, user applications, network prioritisation services, environmental conditions and traffic load of capturing STA [11]. These make this research a good candidate for artificial neural networks (ANN) also commonly known as neural networks (NN). NNs have a very high flexibility and, hence, can analyse incomplete or partial data. However, WLAN traffic and NN parallel processing feature can generate a significant amount of overhead on the monitoring STA, which can affect performance of the monitoring STA, and sometimes can lead to a denial of service (DoS).

In order to classify as a user or attack frame, the research analyses four distinct parameters only. These parameters are sequence number and frame sub-type of a MAC frame; a signal attribute, signal strength indicator (SSI); and statistical information, delta time value. Capturing and processing few parameters have a low impact on the monitoring STA. The preliminary work on selecting these attributes has been published in Ratnayake et al. [12]. The rest of the paper is organised as follows: Sect. 2 reviews the related current work on probing attack detection and prevention using non-intelligent and intelligent methods; Sect. 3 defines the probe request attack detection methodology; Sect. 4 discusses the WLAN organisation, data capturing and preparation methods, NN design, evaluation and results; and Sect. 5 concludes the paper.

2 Literature review

Many researchers have worked in the area of network security looking for possible solutions for intrusion detection, to recognise an adversary attempting to gain access, or have already compromised the computer network [13]. Ratnayake et al. [12] analysed non-intelligent and intelligent wireless intrusion detection systems (WIDS). Non-intelligent methods, also known as conventional methods [2–4, 14–22], lack flexibility to adaptation to environmental changes and therefore become outdated. WLAN security researchers are now gradually moving towards soft-computing techniques [5–7, 23–26]. Some of the popular methods are self-organising maps, artificial immune systems, fuzzy logic and neural models, adaptive neural-fuzzy inference systems and hybrid models. They play a major role in current research due to their capability to overcome many integral weaknesses in conventional intrusion detection systems (IDS) such as adaptability issues, which require frequent updates and high computation power and time. However, soft-computing techniques too suffer from their inherited design weaknesses such as requirement of training data; pre-processing of data, time and computing resources required for training or learning; validation; testing; optimisation and simulation of models.

Further, less consideration is given to the most crucial issue of real-world data collection and real-world application. Training, validating and testing on real data and simulation on real data are very important in the process of bringing the research into real-world applications. Use of existing sample datasets such as KDD Cup ‘99 dataset and SNORT or other sanitised or simulated traffic to develop and test intrusion detection proposals is a popular approach in the current literature. The KDD Cup ‘99 dataset is created by processing the tcpdump portions of the 1998 DARPA IDS evaluation dataset. DARPA normal dataset is a simulated synthetic data, and attack data are generated through scripts and programs. These datasets therefore do not contain information that a real-world dataset consists of [8–10]. SNORT database on the other hand has not been updated since year 2005. These problems also apply to solutions that are based on other sample databases and synthetic or sanitised traffic. Apart from the issues discussed above, traffic generated from test beds also has the issue of limited environmental conditions as data will be collected from a controlled network by simulating traffic. Some solutions identify intrusive behaviours based on the exploration of known vulnerabilities.

Collection and use of real-world traffic also makes the researcher understand the real-world issues that a security administrator may encounter whilst implementing a proposed application. However, real-world data collection can lead to biased data being used for training and testing, as there is no standard approach or guidelines for collecting and using traffic of a real network. Furthermore, as the dataset is unique to each experiment, results cannot compare with other research, unless one implements other methods on the same dataset to compare two methods.

Furthermore, some of the existing studies do not explain how they recognised genuine and attack frames within the training traffic when real-world data are collected [6, 9]. The research assumes that they may have collected attack and normal frames separately, or collect traffic whilst an attacker is available, and analysed manually to label them based on other features.

Many of the existing approaches of intrusion detection have focused on the issues of feature extraction. Selecting input features based on the highest eigenvalue from a limited set of data may lead to losing many important and

Journal : Large 521 Dispatch : 30-8-2013 Pages : 14
Article No. : 1478 LE TYPESET
MS Code : NCA-3163 CP DISK

sensitive features, which can affect the efficiency and effectiveness of the classifier. Almost no research evaluates the results of a detection model's performance to different types of scenarios, e.g. when user and attacker(s) at different distances from AP, when only the attacker is present, when there is no attacker, when there is more than one attacker, when there are attackers with similar and different network interface card (NIC) types, and so on. This lack of information can confuse or mislead readers or future researchers, as although their proposals are excellent in technical and practical aspects, they may not reach the outstanding results that other researches may have published using non-challenging traffic.

Further, most of IDSs propose universal solutions to intrusions. This research agrees with Liao et al. [27] who suggest that the existing IDSs pose challenges to the categories that they claim they belong to. Many are extremely complex proposals that are simulated and tested without considering the practical implementation and computing power, they may be required, therefore limited for academic research world as implementation is too complex or expensive. This issue has also been identified by Liao et al. [27].

There is a broad variety of statistical methods used in the literature for measuring the performance of NNs [28, 29]. However, mean squared error (MSE), regression, confusion matrices and operating characteristic (ROC) curve are the most commonly used methods in the field of intrusion detection using NN. In a previous publication, this research implemented a prototype of the proposed design as a function approximation application [12]. Performance is evaluated using linear regression value R^2 and MSE. The sensor trained outstandingly producing 98 % overall regression, and MSEs 0.0039, 0.0038 and 0.0037 on training, validation and test samples. However, simulation of classifier in eight scenarios with 1,000 frame samples resulted only in an average of 94.5 % detection rate. Conventionally, probe request attack detection is a binary classification application where the output can only have two values, 1 (attack) or 0 (no attack). The research in [12] applied standard linear regression and treated the output as if it is binary, classifying any value of 0.5 or above as a '1', and anything below 0.5, as a '0'. Although standard linear regression has been applied successfully for classification in the past and in current research applications, statisticians argue that linear regression should not be used for binary classification applications, it violates many assumptions of linear regression [30–32].

3 Probe request attack detection methodology

The process of establishing the IEEE 802.11 association during an active scan is presented in Fig. 1. IEEE standard

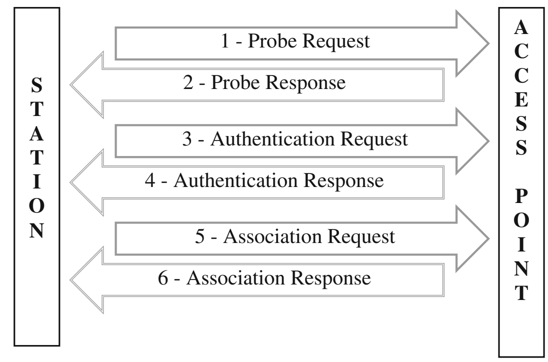

Fig. 1 Active scan and WLAN association

802.11 defines three frame types: management, control and data. The management frames set up and maintain communications. The control frames facilitate in the delivery of data. The data frames encapsulate the open system interconnection (OSI) network layer packets. Each frame consists of a MAC header, frame body and a frame check sequence (FCS); however, contents of frames vary depending on the frame type. Probe requests are management frames and can be sent by anyone with a legitimate MAC address, as association with the network is not required at this stage. A typical management frame header comprises of following: a frame control field that defines the type of MAC frame and information to process the frame; a duration field that indicates the remaining time to receive the next frame; address fields that indicate MAC addresses of destination, source and AP; sequence control information to indicate the sequence number and fragment number of each frame. The frame body contains information specific to the frame type and sub-type. FCS contains an IEEE 32-bit cyclic redundancy check (CRC). Another valuable set of information available to attackers as well as researchers is frame statistical details and radio information generated by the STA that is capturing. This information can be retrieved by using packet analysing software such as Wireshark. Some of the commonly used statistical information in the current research is frame arrival time, time delta value (time since the previous packet is captured), time since a referenced frame, frame number, actual packet length, captured packet length and protocols in frame. In Wireshark frame detail, the IEEE 802.11 radio information is available before the start of the IEEE 802.11 header. This contains signal strength, signal quality (noise), modulation type, channel type, data rate, channel number and other useful information for network and security administrators as well as adversaries [1, 5, 33].

Through these detailed studies, it is learned that before an attack, the attacker actively or passively monitors the network to learn vital network information. MAC address spoofing is the next step. It is therefore recognised that any

Journal : Large 521 Dispatch : 30-8-2013 Pages : 14
Article No. : 1478 LE TYPESET
MS Code : NCA-3163 CP DISK

WIDS should address these initial stages of an attack before moving on to more advance steps. After analysing the previous research work and the progress of IEEE 802.11 sub-committees, it is understood that there is a gap of knowledge to develop a realistic WIDS that could detect probe request attacks on real WLANs. The aim of this research is to provide a flexible, lightweight and low-cost solution that detects an attack during an early stage of the communication with high accuracy and avoid WIDS flaws discussed above.

The research's scope is to detect an external attacker on a single-frequency band of a single AP WLAN. A high computation power is required for a real-world implementation if a solution is to use the full range of fields of a MAC frame, signal attributes and statistical information. Therefore, the research created a short list of attributes shown in Table 1, after studying the IEEE 802.11 specification [1] and predominantly used attributes/features in previous research on DoS attacks on WLANs [2–11, 14–26]. The research then manually refined the list removing features that attackers can easily change, and features which are redundant and dependent, reducing the features to sequence number and frame sub-type of a MAC frame, signal attribute—signal strength indicator (SSI) and statistical information—delta time value to develop a WIDS.

In summary, a rogue STA cannot practically synchronise with a sequence number pattern from a genuine STA. Some signal attributes can be manipulated using identical NICs and configuring accordingly. However, SSI is a nearly impossible feature to mimic. Frame statistics such as frame arrival time is a reliable attribute that attackers cannot manipulate. Delta time gives time difference between two consecutive frames, which is a reliable attribute commonly used by network administrations to review traffic issues. Frame sub-type is a critical attribute identifying a frame type, but can be manipulated by rogue traffic generators and replay attackers; however, the frame subtype manipulation can be detected when combined with other 3 features. Additionally, the research performed a proof-of-the-concept experiment [12] using data captured during an attack from a test bed WLAN. This pilot study provided an opportunity to prove the concepts of IEEE 802.11 standard and to validate many unrealistic concepts based on unwarranted theoretical arguments.

A WIDS should be able to capture and analyse frames and detect attacks automatically in a real network that is unpredictable by nature. After considering different intelligent models and their possible realistic and efficient application on the detection of probe request attacks, the research considered to utilise supervised feed-forward NN architecture. Feed-forward NNs are straightforward networks that associate inputs with outputs, sending signals only in one direction with no feedback loops. Therefore, the output does not affect the same layer. Supervised NNs learn from examples. After training or learning, a NN system is able to detect intrusions, deal with varying nature of attacks. NNs are capable of processing nonlinear data. Therefore, data from several sources can be used in a coordinated fashion to detect attacks. NNs do not need to update frequently, as the generalisation feature enables the

Table 1 Feature selection

Attribute/feature	An attacker can imitate a genuine feature	Can replay attacked	Duplicate/dependent on other attributes
Sequence number	No	Yes	No
Frame type	Yes	Yes	Yes
Frame sub-type	Yes	Yes	No
Duration	Yes	Yes	No
SSID	Yes	Yes	No
FCS	No	Yes	No
Supported data rates	Yes	Yes	No
Protocols in frame	Yes	Yes	Yes
Frame length	Yes	Yes	Yes
Power management	Yes	Yes	No
Frame arrival time	No	No	No
Frame relative arrival time	No	No	Yes
Delta time	No	No	Yes
Frame length captured	No	No	Yes
SSI	No	No	No
Channel type	Yes	No	No
Channel number	Yes	No	No
Data rate	Yes	No	Yes

Journal : Large 521 Dispatch : 30-8-2013 Pages : 14
Article No. : 1478 LE TYPESET
MS Code : NCA-3163 CP DISK

NN to detect unknown and variants of known attacks. Moreover, mostly used WLAN cards today have IEEE 802.11 g and n standard, which have a maximum data transfer rates of 54 and 600 Mbps, respectively. Hence, when the number of participating stations in the WLAN increases, the number of frames to be captured and processed by the WIDS also increases. A NN can also handle a large quantity of data and has very high processing capability due to its parallel processing feature [34–36]. These qualities make NNs a good candidate for detecting WLAN attacks, particularly probe request attacks. However, this solution is limited to detect probe request attacks only whilst a real-world network may experience a cocktail of attacks. This solution also cannot prevent probing attacks and cannot detect any adversary not emitting frames.

Following is the summary of methodology applied for data capturing, training, testing and evaluation of NN (Sect. 4 discusses these methods in detail):

- Apply filtering rules and capture sequence number, delta time, SSI and frame sub-type.
- Capture frames from user and spoofed stations.
- Create master input and target vectors.
- Create sub-input and target vectors (folds) for NN fivefold validation.
- Specify 20 hidden neurons and create NNs using each fold
- Set data division percentages as 70/30 for training and intermediate validation.
- Perform fivefold validation and measure performance using MSE, confusion error and ROC.
- Choose the best performed NN based on confusion error in test phase.
- Simulate trained NN with freshly captured data from the WLAN, which was not used for training the NN or part of the training dataset.
- Analyse performance using classification formulae given in Table 2.

Table 2 Classification formulae [38–40]

$$TN\text{ coverage }\% = \left(\frac{TN}{TN+FP+FN+TP}\right) \times 100 \quad (1)$$

$$FP\text{ coverage }\% = \left(\frac{FP}{TN+FP+FN+TP}\right) \times 100 \quad (2)$$

$$FN\text{ coverage }\% = \left(\frac{FN}{TN+FP+FN+TP}\right) \times 100 \quad (3)$$

$$TP\text{ coverage }\% = \left(\frac{TP}{TN+FP+FN+TP}\right) \times 100 \quad (4)$$

$$TP\text{ rate }\% = \left(\frac{TP}{FN+TP}\right) \times 100 \quad (5)$$

$$TN\text{ rate }\% = \left(\frac{TN}{TN+FP}\right) \times 100 \quad (6)$$

$$FP\text{ rate }\% = \left(\frac{FP}{TN+FP}\right) \times 100 \quad (7)$$

$$FN\text{ rate }\% = \left(\frac{FN}{FN+TP}\right) \times 100 \quad (8)$$

$$+\text{ve Prediction precision }\% = \left(\frac{TP}{TP+FP}\right) \times 100 \quad (9)$$

$$-\text{ve Prediction precision }\% = \left(\frac{TN}{TN+FN}\right) \times 100 \quad (10)$$

$$\text{Accuracy }\% = \left(\frac{TN+TP}{TN+FP+FN+TP}\right) \times 100 \quad (11)$$

$$\text{Confusion }\% = \left(\frac{FP+FN}{TN+FP+FN+TP}\right) \times 100 \quad (12)$$

4 Probe request attack classifier design and evaluation

4.1 WLAN data capture and preparation

A wireless network with 8 user stations is utilised to capture delta time value, sequence number, received signal strength and frame sub-type of the packets transmitted between an AP, users and attackers (Fig. 2).

- AP is a Netgear DG834GT router with MAC address 00:0f:b5:1a:23:82. It is configured with WPA2-PSK enabled controlled access list (CAL), so that only the computers with the listed MAC addresses and network key could access the network resources. AP does not respond to computers with MAC addresses not listed in the CAL. Computers without a network key cannot associate with the AP. However, as shown in Fig. 1, AP replies with probe responses and authentication responses.
- The user station Test1-PC is a DELL Inspiration 510 M laptop with an Intel® Pentium® 1.6 2 GHz microprocessor and 1 MB of random access memory (RAM), with Microsoft Windows XP operating system. Communicates with AP using Intel(R) PRO/WLAN 2100 mini PCI NIC with MAC address 00:0c:f1:5b:dd:b3. Microsoft Office 2007 is the main application software used. IE/Firefox, AVG, Skype, Teamviewer, are some of the other software that are been used.
- Attacker Test2-PC is a Toshiba Satellite Pro laptop with an Intel® Pentium® M 740 (2 GHz) microprocessor and 1.9 gigabytes of RAM, with Microsoft Windows XP. This attacker is spoofed using a commercially available spoofing tool, SMAC 2.0. SMAC 2.0 changes MAC addresses in Microsoft Windows systems, regardless of whether the manufacturers allow this option or not .
- The capturing/monitoring station Test3-PC is a Toshiba Satellite Pro laptop with an Intel® Pentium® M 740 (2 GHz) microprocessor and 1.9 gigabytes of RAM, with BackTrack4 OS. An external network adaptor, Realtek RTL8187 Wireless 802.11 b/g, 54 megabytes, Wireless Universal Serial Bus (USB) 2.0 packet scheduler/mini adaptor, facilitated the monitoring station to be configured to monitor/promiscuous mode in

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | LE | TYPESET
MS Code : NCA-3163 | CP | DISK

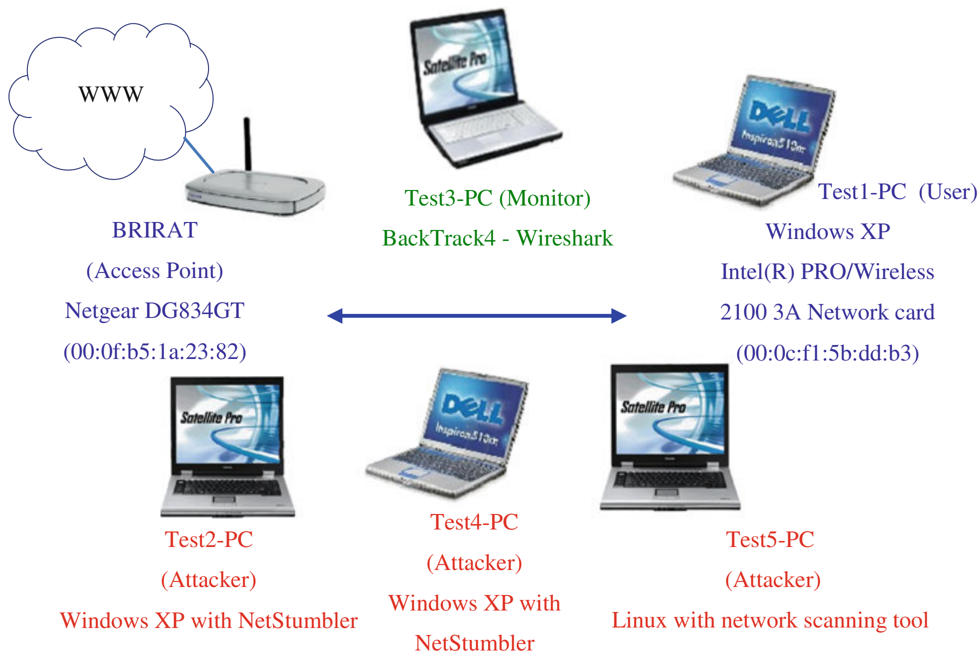

Fig. 2 WLAN

BackTrack environment, so that it receives and reads all data packets transmitted using Wireshark. NIC in promiscuous mode does not emit any frames. Monitoring is restricted to IEEE 802.11 WLAN channel number 11—2,462 MHz due to heavy frame loss experienced when capturing on all channels. Therefore, monitoring statistics for STA's behaviour on the entire bandwidth is unavailable.

The research devised a novel method for capturing data. The research captured frames from user and attacker separately and joined them to create sample training and testing datasets as follows: Data capture is performed in two phases. During the first phase (phase_1), genuine frames are captured from user Test1-PC. The user is asked to note the tasks performed during a specific time period. During this period, the User Test1-PC accessed Internet to browse information, download software, watch a live television channel, listen to a live radio channel and check/send emails. Second phase of the capture started immediately after the first phase. During the second phase (phase_2), the Test1-PC is kept offline. Attacker Test2-PC with its spoofed MAC address is made to send a flood of probe request frames to the AP and made few network key guessing attempts. Both user and attacker performed start-up and shutdown procedures, network scans, network connect and disconnect, and NIC repair (Table 3).

Preliminary checks have been performed to determine that other attackers are not present during the capturing period. A normal Wireshark capture consists of all frames that are received by the NIC of the capturing station. Each frame contains a combination of MAC frame, radio and statistical data. Therefore, filtering rules are applied to

A(a) Filter all frames with source address (wlan.sa) 00:0c:f1:5b: dd:b3
A(b) Filter delta time (frame.time_delta), sequence number (wlan.seq), SSI (radiotap.dbm_antsignal) and frame sub-type (wlan.fc.subtype) of each frame

Phases 1 and 2 consisted of 157,060 and 19,570 frames, respectively.

4.2 NN classifier training and evaluation

In order to detect probe request attacks, a supervised feed-forward NN with 4 input neurons (deltaTime, sequenceNumber, signalStrength and frameSubtype), 1 hidden layer with 20 neurons and an output neuron that determines genuine frames (0) from rogue frames (1) is implemented using MATLAB technical computing language. There are many conventional and modern theories and practices one can implement when determining the number of hidden neurons and layers [37]. A single layer is selected to reduce the complexity of the NN. The research trained the sample dataset with 1–50, step 5, neurons and identified the NN with 20 neurons is the best-performing NN based on MSE and convergence time. The network is trained using scaled conjugate gradient (trainscg) back-propagation function. This function is memory efficient and converges slowly. During training, the NN updates weight and bias values according to the following training parameters: maximum

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | LE | TYPESET
MS Code : NCA-3163 | CP | DISK

Table 3 User and attacker activities

Data segments included	Starting record no.	Action
User activities (Phase_1)		
k1	1	Capture started
k1	315	Opened IE
k1	409	Googgled and played BBC2 live and stopped
k1	1901	Googgled and played BBC1 radio live
k1	7721	Checked Yahoo email
k1	21474	Sent an email
k1, k2, k3, k4, k5	22840	d/loaded a large file
K5	153240	Stopped BBC1 radio
k5	154670	d/load completed
k5	154686	Closed all opened windows
k5	154734	Disconnected from AP
k5	154769	Scanned network
k5	154860	Tried to connect to the network 3 times
k5	155363	Repaired the adaptor
k5	155640	Opened IE
k5	157001	Shut down
	NIL	Stopped capture
Attacker activities (Phase_2)		
	NIL	Capture started
y1	1	Attacker started
y1	7	Directed probing attack started
y1	1577	Directed probing attack stopped
y1	1710	Network scanned 3 $\times$ times
y1	1977	Tried to connect to the network with a guessed network key
y1	1846	Tried to connect to the network with a guessed network key
y1	1871	Tried to connect to the network with a guessed network key
y1	2223	Tried to connect to the network with a guessed network key
y1, y2, y3, y4, y5	2223	Directed probing attack started
y5	18941	Directed probing attack stopped
y5	19148	Network scanned 5 $\times$ times
y5	19490	Tried to connect to the network with a guessed network key
y5	19551	Turned off the attacker
	Nil	Stopped capture

number of epochs = 100, MSE goal = 0, maximum time to train = infinity, minimum performance gradient = 1e − 6, maximum validation failures = 5, second derivative approximation value = 5.0e − 5, parameter for regulating the indefiniteness of the Hessian = 5.0e − 7. It uses tan-sigmoid transfer function in both hidden and output layers as it scales the output values from zero to one.

One of the most common performance measurement methods in use for evaluating a NN designed for classification is MSE. MSE is the average squared error between the NN's output and the target value of a complete data sample. MSE = 0 means no errors. Values closer to 0 are better. This research uses the MSE to evaluate and compare how the NN has learned the training data. After training, testing dataset is passed through the classification system. However, MSE does not give a clear picture of how a model classified its frames. A basic confusion matrix gives sums of correct and incorrect classifications based on true positive (TP), true negative (TN), false positive (FP) and false negative (FN). These results can be further explained using formulae such as TN, FP, FN and TP coverage, and rate percentages, positive and negative prediction precision percentages, accuracy and confusion presented in Table 2.

A series of FP and TP pairs plots a ROC. A ROC is a visual tool to recognise the positive and negative samples

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | LE | TYPESET
MS Code : NCA-3163 | CP | DISK

that are incorrectly identified. When (0.1), the FP = 0 and TP = 1, which indicates a perfect predictor. Therefore, the more each curve hugs the left and top edges of the graph, the better the prediction. The area beneath the curve can be used as a measure of accuracy. ROC also encapsulates all the information presented in a confusion matrix and therefore commonly used by the researchers to show the consistency of results [38, 41, 42].

However, a model's performance can be misleading due to over-fitting, which generally occurs when the model training is not evaluated during the training process. Overfitted models do not perform well on unseen data. Typically, over-fitted models can be recognised from smaller training confusion and larger testing confusion. This issue is addressed by means of intermediate validation during training. Self-consistency and cross-validation are among several methods of estimating how well a trained model will perform with unseen data, and detect and prevent overfitting of the model. Self-consistency is a method to evaluate the model's performance with seen data. In self-consistency test method, frames from phase_2 append to the frames from phase_1 to use as the data sample (FoldAll), and complete dataset (176,630 frames) is utilised for training (70 %) and validation (30 %) phase. Therefore, there is no wastage of training data. During the test phase, the complete dataset (176,630 frames) is reused. However, as the parameters of the NN are obtained from the training dataset, error rate can be underestimated leading to a high accuracy rate. Self-consistency test method does not require much computation as training, validation and testing are executed only once [43, 44].

In order to minimise bias present within the random sampling of the data samples, K-fold cross-validation methodology is used. Here, the original sample is partitioned into k sub-samples. Then, the results from each fold are averaged to generate a single estimation. Tenfold cross-validation is most commonly used to reduce the wastage of data in circumstances where there is a limited set of data. However, when larger numbers of folds are applied to a high data volume, it requires extra computations and processing and is time-consuming [45]. As this research has a large quantity of data, it is decided to use fivefold cross-validation. The frames of phase_1 and phase_2 are divided into 5 equal segments, as shown in Table 4 and labelled as rogue or genuine, for the fivefold cross-validation.

The cross-validation process is repeated 5 times. Each of the 5 sub-samples is used only once as the validation data, i.e. each time a single sub-sample is retained to test the model, whilst remaining 4 sub-samples are used as training data. The system randomly divides the data sample and uses 70 % of the data to train the network and 30 % for validation. The MSEs of self-consistency (FoldAll) and Fold1 to Fold5 test are shown in Table 5.

Table 5 shows in a nutshell that the results are very much similar in every test. However, to understand and analyse the behaviours of MSE and confusion errors of cross-validation and self-consistency test, Figs. 3 and 4 are produced. In Fg. 4, Fold1 shows the best MSEs 0.0022 and 0.0018 for training and validation, respectively. However, Fold1 test records the worst MSE 0.0167. Further, it shows that Fold1 significantly deviates from the rest of the folds. The worst MSE during training is generated by Fold5. Fold4 shows the worst MSE during validation and best MSE during test. Further, it also shows that MSEs of test are higher than the training and validation in Fold1 and Fold2, which indicates an over-fit. Figure 4 shows that the confusion percentages of the classifiers are extremely low, resulting in an accuracy rate ranging from 98.19 to 99.88 % during training, validation and test. Fold1 shows the lowest confusion rates 0.22 and 0.19 % for training and validation, respectively. However, Fold1 test records the highest confusion 1.81 %. Further, it shows that Fold1 relatively significantly deviates from the rest of the folds. The highest confusion during training is 0.59 %, generated in Fold5. The highest confusion during validation is generated in Fold4. The least confusion during test is 0.12, produced in Fold4. Further, it also shows that confusions of tests are higher than the training and validation in Fold1 and Fold2. Both MSE and confusion values of self-consistency test have a clear least deviation among training, validation and test results: self-consistency test reports MSEs as 0.0043, 0.0045, 0.0043 (Fig. 3) and confusions, as 0.47, 0.50, and 0.47 % (Fig. 4). The ROC curves in Fig. 5 are a graphical representation of sensitivity and specificity. It also visually summarises the results of fivefold cross-validation (Fold1–Fold5) tests and self-consistency test (FoldAll) presented in the confusion matrix. Figure 6 is a cross-section of Fig. 5. In the graph, all curves hug the left and top edges of the plot, which proves that the trained NNs are nearly perfect predictors. Further, the area beneath the curves also shows a high measure of accuracy.

In summary, Fold1 and Fold2 have a risk of overfitting, as its test confusion is greater than the validation confusion. From the remaining Folds3–5, Fold4 with test confusion rate 0.12 % has the least confusion, therefore becomes the best-performing model. Therefore, Fold4 qualifies to simulate with unseen data. The information in Table 6 is obtained by applying TP, TN, FP and FN results attained from the Fold4 test phase to classification formulae in Table 2.

The overall analysis of these results in Table 6 shows that there are no major deviations in results that is generalised, when calculating confusion or accuracy. The confusion percentage is extremely low, resulting in an accuracy rate of 98.5 % on unseen test dataset. This indicates that the classifier's performance is nearly perfect.

Table 4 Data segmentation method

Data segments (k + y)	Genuine frames from normal user (Phase_1)		Rogue frames from attacker (Phase_2)		Running total
	Start	End	Start	End	
k1 + y1	0	31,412	0	3,914	35,326
k2 + y2	31,413	62,824	3,915	7,828	70,652
k3 + y3	62,825	94,236	7,829	11,742	105,978
k4 + y4	94,237	125,648	11,743	15,656	141,304
k5 + y5	125,649	157,060	15,657	19,570	176,630

Table 5 MSE and confusion errors of self-consistency and fivefold validation

Description	FoldAll	Fold1	Fold2	Fold3	Fold4	Fold5
Train sample	1,2,3,4,5	1,2,3,4	2,3,4,5	3,4,5,1	4,5,1,2	5,1,2,3
Test sample	1,2,3,4,5	5	1	2	3	4
MSE						
Training	0.0043	0.0022[a]	0.0039	0.0044	0.0051	0.0053[b]
Validation	0.0045	0.0018[a]	0.0045	0.0045	0.0052[b]	0.0049
Test	0.0043	0.0167[b]	0.0058	0.0018	0.0012[a]	0.0014
Confusion						
Training (%)	0.47	0.22[a]	0.42	0.49	0.56	0.59[b]
Validation (%)	0.50	0.19[a]	0.49	0.50	0.58[b]	0.54
Test (%)	0.47	1.81[b]	0.68	0.18	0.12[a]	0.13

[a] Lowest value

[b] Highest value[h]

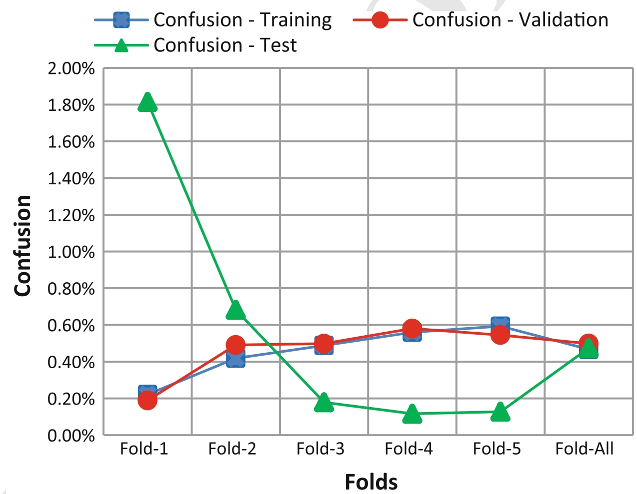

Fig. 4 Confusions of cross-validation and self-consistency tests

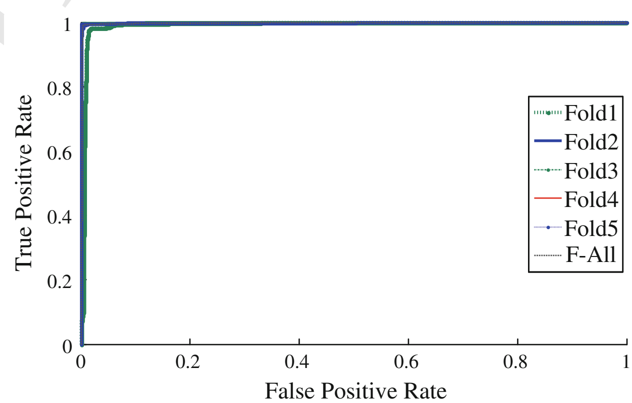

Fig. 5 ROC curves of Fold1 to Fold5 and self-consistency tests

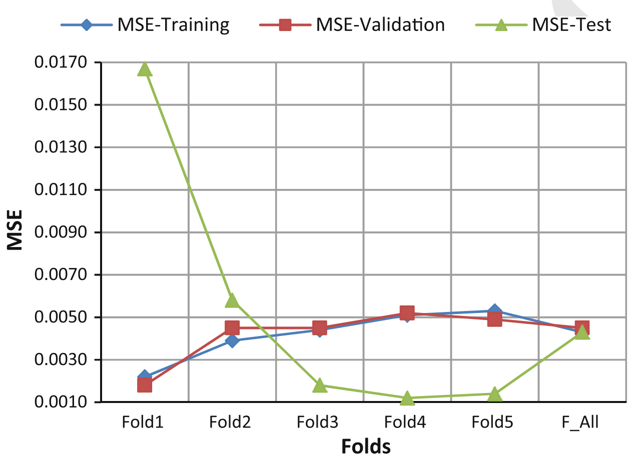

Fig. 3 MSEs of cross-validation and self-consistency tests

Furthermore, high sensitivity and specificity rates prove the robustness and stability of the NN model. The Fold4 ROC presented in Figs. 5 and 6 hugs the left and top edges of the plot, graphically proves the consistency of the NN and the area beneath the curve illustrates high measure of accuracy.

To understand the results in Tables 5 and 6, and in Figs. 3, 4, 5 and 6, the user and attacker activities during the capturing period presented in Table 3 are analysed with data segments used for NN training, validation and testing (Table 4). Fold5 uses data segments 1, 2, 4 and 5 to train the network and leaves segment 3 to test the network. The analysis in Table 3 indicates that the training sample with segments 1, 2, 4 and 5 is diverse. It also shows that trained NN performs considerably well with unseen data.

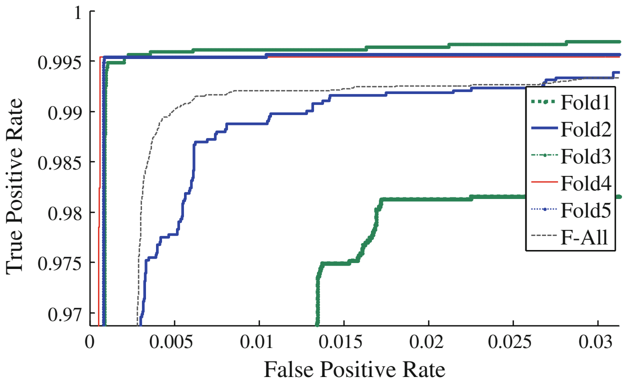

Fig. 6 A cross-section of ROC curves

Table 6 Confusion matrix of Fold4 test

Description	Fold4
Training data segments (ky)	4,5,1,2
Test data segment	3
Testing sample	35,326
TP	3,892
TN	31,393
FP	19
FN	22
TP coverage	11.02 %
TN coverage	88.87 %
FP coverage	0.05 %
FN coverage	0.06 %
Sensitivity TP	99.44 %
Specificity TN	99.94 %
FP	0.06 %
FN	0.56 %
Positive prediction precision	99.51 %
Negative prediction precision	99.93 %
Accuracy	99.88 %
Confusion	0.12 %

4.3 Simulation results and discussion

In addition to the attacker Test2-PC utilised in the training capture, two new attackers are utilised for simulation, namely Microsoft Windows-based Test4-PC and Linux-based Test5-PC.

- The attacker station Test4-PC is a DELL Inspiron 510 M laptop identical to TEST1-PC. This attacker is spoofed using SMAC 2.0.
- Attacker Test5-PC is a Toshiba Satellite Pro laptop with an Intel® Pentium® M 740 (2 GHz) microprocessor and 1.9 gigabytes of RAM, with Linux-based Ubuntu 9.10 OS. NIC is Netgear WG111T 108 Mbps USB 2.0 Adapter. This attacker is spoofed using macchanger spoofing tool.

Frames are captured using the capturing STA Test3-PC. Capturing sessions varied to collect adequate number of frames, approximately 3,000 frames per session. Traffic captured from user STAs is normal uncontrolled traffic. However, the research generated the probe request attacks using NetStumbler and Linux network scanning tool. Trained NN is simulated using the pre-defined scenarios as shown in Table 7. Frames are captured by using Wireshark capturing software with same filtering rules, A(a) and A(b), when capturing training data. Data are captured in numerical form. Therefore, no conversion is needed.

Figure 7 shows security administrator's probe request attack monitoring screen. The system tabulates classifier's output values against frame numbers. Output bounds are (0,1). Ideally, the output value should be zero, which means 'no attack'. There are many schools of thought as to how one classifies a frame into an attack or genuine class. This research uses the most common method, that is, frames with output neuron value equal or higher than 0.5 are classified as attack or positive frames (1), whilst others are classified as genuine or negative frames (0). However, in real-world situations, security administrators can set the threshold value depending on the degree of sensitivity required. A real-world application also can provide more information on the screen such as the MAC address, time and other statistics. However, this research cannot verify the accuracy of the detection system from Fig 7. Therefore, result validation requires comparing the actual results with expected results.

The Fig. 8 tabulates the squared difference between the expected value (target) and the actual result (output) of each frame of the complete dataset, and produces an error value scaled from zero to one. A frame's error = zero means 'no error', that frame is correctly classified. The MSE of the dataset is 0.034262, which is a value very close to zero that is statistically a good performance. This also generates 4.1 % overall confusion resulting 95.9 % of overall accuracy from the 10 simulation samples used in Table 7. The individual simulation results of pre-defined scenarios shown in Table 7 are tabulated in Table 8.

Following is the interpretation of the results of Table 8.

4.3.1 Detection rate of known and unknown attacks

This research refers the NetStumbler attack frames that the NN is trained with as known attacks. Unknown attacks are frames generated from software that were not used for training the NN. The results of simulations 34, 36–38 show that the NN has detected between 99.66 and 100 % of known NetStumbler attacks. The results of simulation 40

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | LE | TYPESET
MS Code : NCA-3163 | CP | DISK

Table 7 Tests conducted

Sim code	Test scenario
Sim31	Unseen dataset from user (Test1-PC)
Sim39	Unseen dataset from user (Test1-PC) far away from AP
Sim34	Unseen dataset from attacker (Test2-PC) using NetStumbler
Sim35	Unseen dataset with attacker (Test2-PC) at the same location as the user
Sim36	Unseen dataset with attacker (Test1-PC) far away from user's location
Sim37	Unseen dataset with new attacker (Test4-PC) using NetStumbler
Sim38	Unseen dataset with 2 attackers (Test2-PC and (Test4-PC) using NetStumbler
Sim40	Unseen dataset with an attacker using a Linux network scanning tool (Test5-PC)
Sim41	Unseen dataset from user (Test1-PC) and attacker (Test2-PC)using NetStumbler
Sim42	Unseen dataset from user (Test1-PC) and an attacker using a Linux network scanning tool (Test5-PC)

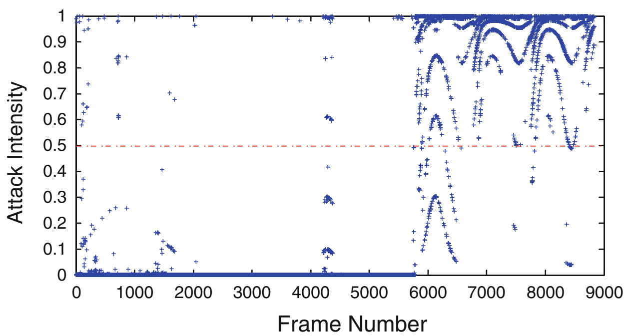

Fig. 7 Security administrator's probe request attack monitoring screen

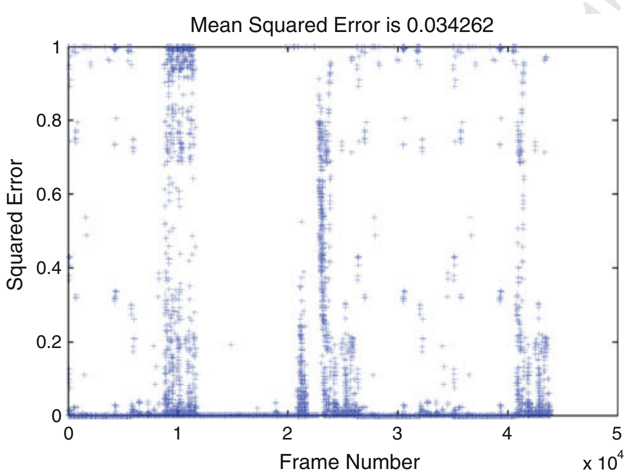

Fig. 8 MSE of overall simulation

show that NN has detected 95.89 % of unknown Linux network scanning tool attacks. The accuracy of trained NN was 99.88 % (Table 6). Therefore, whilst the detection rates of known attacks (sim34, 36–38) are 99.5, 100, 100 and 100 %, respectively, the detection of unknown attacks shows 4.11 % reduction.

4.3.2 Effect of the movement of an attacker and a user

It was observed that when the NetStumbler attacker was at a general distance or far away location within the signal range, the detection rate was 99.66–100 % (sim34 and 36). When the NetStumbler attacker was at the same locations as the user, the detection rate was only 72.91 % (sim35). However, it is a nearly impossible scenario in a non-public WLAN. The results of sim39 showed that the detection rate reduces when user moves away from the signal range. When the captured data are analysed, it is observed that when a genuine user scans a network excessively, it can raise a false alarm, because it generates unusually a large number of probe requests. This can occur due to an ill-configured WLAN card, weak signal strength or as in this case, user deliberately scanning the network. This may require network administrator's attention and can be solved within the system by setting a threshold value of warnings to be tolerated per second to suit to specific users or network. However, user's mobility within the signal range does not affect the detection rate very much, and therefore, this solution enables the WLAN users to change their location of work in contrast to some experiments that required user stations to be static.

4.3.3 Effect of the user and an attacker's presence at the same time

The NN was simulated (sim41) using a random combination of data used for sim31 and sim34, which is an unseen dataset from user and attacker using NetStumbler (known attack). In this scenario, sensitivity and specificity of the scenario was 98.18 and 99.66 % respectively, which was similar to sensitivity and specificity of sim31 and sim34. However, there is a reduction in positive prediction rate from 100 to 96.58 % and negative prediction rate from 100 to 99.82 % in sim41. Further, it reports a 1.31 % of

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | LE | TYPESET
MS Code : NCA-3163 | CP | DISK

Table 8 Summary of tests conducted

Description	Sim31	Sim39	Sim34	Sim35	Sim36	Sim37	Sim38	Sim40	Sim41	Sim42
Complete sample	5,782	3,595	2,975	2,905	2,837	2,026	3,088	3,045	8,827	8,757
TP	0	0	2,965	2,118	2,837	2,026	3,088	2,920	2,965	2,920
TN	5,677	3,180	0	0	0	0	0	0	5,677	5,677
FP	105	415	0	0	0	0	0	0	105	105
FN	0	0	10	787	0	0	0	125	10	125
TP coverage (%)	0.00	0.00	99.66	72.91	100.00	100.00	100.00	95.89	33.86	33.08
TN coverage (%)	98.18	88.46	0.00	0.00	0.00	0.00	0.00	0.00	64.83	64.31
FP coverage %	1.82	11.54	0.00	0.00	0.00	0.00	0.00	0.00	1.20	1.19
FN coverage (%)	0.00	0.00	0.34	27.09	0.00	0.00	0.00	4.11	0.11	1.42
Sensitivity TP (%)	NaN	NaN	99.66	72.91	100.00	100.00	100.00	95.89	99.66	95.89
Specificity TN (%)	98.18	88.46	NaN	NaN	NaN	NaN	NaN	NaN	98.18	98.18
False +ve discovery rate (%)	1.82	11.54	NaN	NaN	NaN	NaN	NaN	NaN	1.82	1.82
False −ve discovery rate (%)	NaN	NaN	0.34	27.09	0.00	0.00	0.00	4.11	0.34	4.11
+ve Prediction precision (%)	0.00	0.00	100.00	100.00	100.00	100.00	100.00	100.00	96.58	96.53
−ve Prediction precision (%)	100.00	100.00	0.00	0.00	NaN	NaN	NaN	0.00	99.82	97.85
Accuracy (%)	98.18	88.46	99.66	72.91	100.00	100.00	100.00	95.89	98.69	97.39
Confusion (%)	1.82	11.54	0.34	27.09	0.00	0.00	0.00	4.11	1.31	2.61

Some formulae generate NaN values when either user or attack frames are presented for calculations where calculations require both user and attack frames

confusion, which is a rate higher than sim34, less than sim31. sim42 utilised a random combination of data used for sim31 and sim40, which is an unseen dataset from user and an attacker using a Linux network scanning tool (unknown attack). In this scenario too, the sensitivity and specificity of the model was 95.89 and 98.18 % respectively, which was similar to sensitivity and specificity of sim31 and sim40. Again, there is a reduction in positive prediction rate from 100 to 96.53 % and negative prediction rate from 100 to 97.85 % in sim42. Further, it reports a 2.61 % of confusion, which is a rate less than sim40 and higher than sim31. It is clear that confusion rate slightly increases during an unknown attack. However, this experimentation shows that the model could still detect an unknown attack with 97.39 % accuracy.

5 Conclusion

This experimental study is carried out to detect probe request attacks by analysing real WLAN traffic frames of a STA using a NN classifier. The supervised feed-forward NN classifier analyses four distinct parameters such as delta time, sequence number, signal strength and frame sub-type, and identify and differentiate a genuine frame from a rogue one. Currently, identifying genuine and rogue frames from real-world traffic for NN training is conducted manually, which is labour-intensive. The proposed solution enables security administrators to train the NN with a diverse combination of separately captured genuine user data and rogue attacker data when necessary. The experimental results demonstrate that the NN model can detect probe request attacks to a very high degree. The proposed solution detects an attack during an early stage of the communication. The solution also works when network prioritisation services like quality of service (QoS) is enabled and works well when the genuine user is offline. Furthermore, although the detection rates slightly drop when STAs move to boundaries of the network, the solution does not limit the genuine STA's movement within the network. Monitoring only delta time, sequence number, signal strength indicator and frame sub-type considerably reduces the overhead of the monitoring machine, whilst producing the expected results as all four fields are nearly impossible to manipulate at any one time. Therefore, this is an efficient, lightweight and low-cost solution, compared to solutions currently available, which needs capturing and processing STAs with high computing power. Furthermore, this may also ease the housekeeping of training data, as administrators can remove unnecessary parts of training data easily and add new training data without having to recapture already available data in circumstances such as replacing or upgrading a STA. This research, by design, is limited to a single-frequency band of a single AP WLAN and can only detect an external attacker. However, the applicability of this research can be improved including features relevant to channel and AP. More research has to be done to improve detection rates when STAs are very

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | □ LE | □ TYPESET
MS Code : NCA-3163 | ☑ CP | ☑ DISK

close to the AP and far away from APs, and to understand the issues of updating the NN with new attack types and user scenarios.

References

1. IEEE (2012) 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 (Revision of IEEE Std 802.11-2007). doi:10.1109/ieeestd.2012.6178212
2. Bernaschi M, Ferreri F, Valcamonici L (2008) Access points vulnerabilities to DoS attacks in 802.11 networks. Wireless Netw. doi:10.1007/s11276-006-8870-6
3. Me G, Ferreri F (2009) New vulnerabilities to DoS attacks in 802.11 networks. http://www.wi-fitechnology.com/Papers+req-showcontent-id-5.html. Accessed 28 July 2009
4. Bicakci K, Tavli B (2009) Denial-of-service attacks and countermeasures in IEEE 802.11 wireless networks. Comput Stand Interfaces. doi:10.1016/j.csi.2008.09.038
5. Ahmad I, Abdullah AB, Alghamdi AS (2009) Application of artificial neural network in detection of probing attacks. IEEE. doi:10.1109/ISIEA.2009.5356382
6. Ataide RLR, Abdelouahab Z (2010) An architecture for wireless intrusion detection systems using artificial neural networks. Novel Algorithms Techn Telecommun Netw. doi:10.1007/978-90-481-3662-9_61
7. He C, Mitchell JC (2005) Security analysis and improvements for IEEE 802.11i. In: Proceedings of 12th annual network and distributed system security symposium, San Diego, CA, pp 90–110
8. McHugh J (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory, TISSEC. doi:10.1145/382912.382923
9. Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J (2003) A comparative study of anomaly detection schemes in network intrusion detection. Society for Industrial & Applied, pp 25–36
10. Lippmann RP, Fried DJ, Graf I, Haines JW, Kendall KR, McClung D, Weber D, Webster SE, Wyschogrod D, Cunningham RK (2000) Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. doi:10.1109/DISCEX.2000.821506
11. Guy CG (2006) VoIP over WLAN 802.11b simulations for infrastructure and ad-hoc networks. In: Proceedings of London communications symposium (LCS 06), pp 61–64, London, UK
12. Ratnayake D, Kazemian H, Yusuf S, Abdullah A (2011) An intelligent approach to detect probe request attacks in IEEE 802.11 networks. Eng Appl Neural Netw. doi:10.1007/978-3-642-23957-1_42
13. Karygiannis T, Owens L (2002) Wireless network security. NIST Spec Publ 800:48
14. Bansal R, Tiwari S, Bansal D (2008) Non-cryptographic methods of MAC spoof detection in wireless LAN. ICON. doi:10.1109/ICON.2008.4772621
15. Malekzadeh M, Azim A, Zulkarniam Z, Muda Z (2007) Security improvement for management frames in IEEE 802.11 wireless networks, International Journal of Computer Science and. Netw Secur 7(6):276–284
16. Madory D (2006) New methods of spoof detection in 802.11 b wireless networking, Thayer School of Engineering, Dartmouth College, Hanover, New Hampshire
17. Qing L, Trappe W (2007) Detecting spoofing and anomalous traffic in wireless networks via forge-resistant relationships. IEEE Trans Forensics Secur. doi:10.1109/TIFS.2007.910236
18. Goel S, Kumar S (2009) An improved method of detecting spoofed attack in wireless LAN. Netw Commun. doi:10.1109/NetCoM.2009.75
19. Lim YX, Schmoyer T, Levine J, Owen HL (2004) Wireless intrusion detection and response: a classic study using main-in-the-middle attack. Wireless Commun. doi:10.1109/WCNC.2004.1311303
20. Guo FL, Chiueh TC (2006) Sequence number-based MAC address spoof detection. Recent Adv Intrus Detect. doi:10.1007/11663812_16
21. Faria DB, Cheriton DR (2006) Detecting identity-based attacks in wireless networks using signalprints. In: Proceedings of 5th ACM workshop on wireless security. doi:10.1145/1161289.1161298
22. Bellardo J, Savage S (2003) 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In: Proceedings of 12th USENIX security symposium, Washington, DC, USA
23. Pleskonjic D (2003) Wireless intrusion detection systems (WIDS). In: Proceedings of 19th annual computer security applications conference, Las Vegas, Nevada, USA
24. Yang H, Xie L, Sun J (2004) Intrusion detection solution to WLANs. In: Proceedings of the IEEE 6th circuits and system symposium on emerging technologies: frontiers of mobile and wireless communication. doi:10.1109/CASSET.2004.1321948
25. Dasgupta D, Gomez J, Gonzalez F, Kaniganti M, Yallapu, K (2003) MMDS: multilevel monitoring and detection system. In: Proceedings of 15th annual computer security incident handling conference, Ottawa, Canada
26. Sheikhan M, Jadidi Z, Farrokhi A (2012) Intrusion detection using reduced-size RNN based on feature grouping. Neural Comput Appl. doi:10.1007/s00521-010-0487-0
27. Liao HJ, Tung KY, Richard Lin CH, Lin YC (2012) Intrusion detection system: a comprehensive review. J Netw Comput Appl. doi:10.1016/j.jnca.2012.09.004
28. Sokolova M, Lapalme G (2009) A systematic analysis of performance measures for classification tasks. Inf Process Manag. doi:10.1016/j.ipm.2009.03.002
29. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
30. Statgun (2007) Logistic regression tutorial. http://www.statgun.com/tutorials/logistic-regression.html. Accessed 05 Nov 2010
31. Ciuiu D (2008) Pattern classification using polynomial and linear. In: Proceedings of international conference trends and challenges in applied mathematics, pp 153–156
32. Peng CYJ, Lee KL, Ingersoll GM (2002) An introduction to logistic regression analysis and reporting. J Educ Res 96(1):3–14
33. Orebaugh A, Ramirez G, Burke J (2007) Wireshark & ethereal network protocol analyzer toolkit. Syngress, USA
34. Stergiou C, Siganos D (1996) Neural networks, vol 2012. Imperial College, London
35. Coolen ACC (2010) A beginner's guide to the mathematics of neural networks. Concept Neural Netw. doi:10.1007/978-1-4471-3427-5_2
36. Haykin S (1994) Neural networks: a comprehensive foundation. MacMillan College, USA
37. Sarle WS (2001) How many hidden units should I use? http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-10.html. Accessed 5 June 2012\
38. Hamilton HJ (2012) Computer science 831: knowledge discovery in databases. http://www2.cs.uregina.ca/~hamilton/courses/831/index.html. Accessed 9 Feb 2013
39. Van Trees HL (2001) Detection, estimation, and modulation theory: part 1, detection, estimation, and linear modulation theory. doi:10.1002/0471221082

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | LE | TYPESET
MS Code : NCA-3163 | CP | DISK

40. MathWorks (2012) Plot classification confusion matrix—plotperform. http://www.mathworks.co.uk/help/toolbox/nnet/ref/plotconfusion.html. Accessed 5 Sept 2012
41. MathWorks (2012) roc—receiver operating characteristic. http://www.mathworks.co.uk/help/toolbox/nnet/ref/roc.html. Accessed 15 May 2012
42. Zaknich A (2003) Neural networks for intelligent signal processing. World Scientific, Singapore
43. Kumar M, Gromiha M, Raghava G (2007) Identification of DNA-binding proteins using support vector machines and evolutionary profiles. BMC Bioinfor. doi:10.1186/1471-2105-8-463
44. MathWorks (2012) Neural network toolbox—crab classification demo. http://www.mathworks.co.uk/products/neural-network/examples.html?file=/products/demos/shipping/nnet/classify_crab_demo.html. Accessed 1 Sept 2012
45. Moore AW (2012) Cross-validation for detecting and preventing overfitting. http://www.autonlab.org/tutorials/overfit10.pdf. Accessed 10 May 2012

Journal : Large 521 | Dispatch : 30-8-2013 | Pages : 14
Article No. : 1478 | LE | TYPESET
MS Code : NCA-3163 | CP | DISK

Journal : **521**
Article : **1478**

Author Query Form

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author

During the process of typesetting your article, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the 'Author's response' area provided below

Query	Details Required	Author's Response
AQ1	Figure 3 is given in editable mode so Figure 3 is processed as Table 2. Remaining figures and tables are renumbered respectively. Please check and confirm.	