ABSTRACT

This paper proposes WIDS, a wireless intrusion detection system, which applies data mining clustering technique to wireless network data captured through hardware sensors for purposes of real time detection of anomalous behavior in wireless packets. Using hardware sensors to capture network packets enables detection of attacks before they reach access points and ensures all packets transmitted in the networks are analyzed for a more complete attack detection. The proposed mining based technique for wireless network intrusion detection contributes by reducing the need for training data, reducing false positives and increasing the effectiveness of attack detection on networks with few (one to twenty) connections.

The proposed WIDS design approach involves real time pre-processing of sensor data using a density-based, Local Sparsity Coefficient (LSC) outlier detection algorithm to assign anomaly scores to the connection records. Connection records with low anomaly scores are used as initial starting cluster centre positions for building clusters. The algorithm continuously derives minimum deviation as the maximum of distances between all pairs of cluster centre positions. New records which have their distances from the closest cluster more than the minimum deviation, are tagged as anomaly and moved to alert cluster. One major result of this paper is detection of MAC spoofing attacks by tracking sequence numbers, which ensures duplicate or spoofed (stolen) MAC addresses are not used in the network.

Categories and Subject Descriptors
H.2.8 [Database Management]: [Database Applications, Data Mining]; K.6.5 [Management of Computing and Information Systems]: Security and Protection—Invasive software, Unauthorized access

General Terms
Wireless intrusion detection

Keywords
hardware sensor, clustering, wireless attacks, CommView for WIFI

1. INTRODUCTION

Intrusion attacks can result in loss of important and confidential data or information that may have disastrous effect on businesses or individuals. A typical example of wireless attack is data or file theft [11] by hackers, through unauthorized access to wireless network. This unauthorized access can be done by decrypting the wired encryption privacy (WEP) key for securing wireless networks by using a war driving software like Netstumbler. Once the WEP key is decrypted, the hacker places illegal wireless access point (WAP). With the created WAP, the network is compromised and the attacker has access to corporate network. In addition to different intrusion prevention practices, such as data protection (e.g., encryption), user verification (e.g., password, fingerprints, and biometrics), an intrusion detection system can also be used as another barrier to guard computer systems. An Intrusion Detection System (IDS) is any system that is able to detect non-permitted deviations (or security violations) [2]. An IDS can be either host-based for monitoring system calls or logs, or network-based for monitoring the flow of network packets. Modern IDSs are usually a combination of these two approaches. Traditional IDSs [2] are also classified as misuse or anomaly detection systems. While misuse detection systems (e.g., Norton antivirus) identify patterns of traffic or application data presumed to be malicious using well-known attacks or weak points of the system, anomaly detection systems (e.g., common Firewalls) compare activities against a ‘normal’ baseline. One problem with the misuse detection systems is that it is not capable of detecting new attacks, unlike anomaly detection systems which are capable of detecting new attacks. However, the main problem with these traditional IDS’s is that they need human intervention. Security ex-
perts spend a lot of time analyzing all known intrusions using their domain knowledge and hand-coding signatures (patterns) for those intrusions. Another issue is that these IDS’s need to be updated whenever new intrusions become known, otherwise they are not able to detect the intrusion. The use of wireless links renders the network susceptible to attacks ranging from passive eavesdropping to active interfering. Unlike wired networks, where an adversary must gain physical access to the network wires or pass through several lines of defense (firewalls and gateways), attacks on a wireless network can come from all directions and target any node. Wireless networks are not only susceptible to TCP/IP-based attacks (such as IP Spoofing, TCP Session Hijacking, SYN Flood) native to wired networks, they are also subject to a wide array of 802.11-specific threats. The IEEE 802.11 specifications represent an over-the-air inter-
face between a wireless client and a base station or between two wireless clients.

1.1 Wireless LAN Architecture and Attacks

Wireless Local Area Network (WLAN) is based on IEEE 802.11 standards. The 802.11 applies to the lower layers, namely Data Link Layer and Physical Layer (PHY) of the wireless open system interconnect (OSI) network model. WLAN’s allow for three different ways to configure a network: ad-hoc or Independent Basic Service Set (IBSS), infrastructure or Extended Service Set (ESS), and a mixture of both or Basic Service Set (BSS). In ad-hoc network, connection is established for the duration of one session and requires no base station (computer). Computers (or stations, e.g., laptops) discover others within range and establish communication. A wireless access point (WAP or AP) is a device that connects wireless communication devices together to form a wireless network. When a WAP is introduced in IBSS, the network becomes BSS and the AP acts as a master to control the stations within that BSS.

Stations that want to join a Basic Service Set (BSS) have to first authenticate themselves and then associate themselves with the BSS. The association will let the station know what transmission rate(s) are available as well as other parameters of the BSS. When the station wants to leave BSS, it will disassociate from the BSS and if it wants to join again, it has to re-authenticate and re-associate with the AP. The two authentication mechanisms provided in WLAN are open authentication and shared key authentication, which require clients to provide a secret key to authenticate to an AP by communicating with frames.

Format of a Wireless Packet

Communication between stations and AP occur through packets (or frames). A typical wireless packet has the following information for data attributes:
1. Wireless Packet Information (this has about 6 sub fields like signal level, noise level, etc.)
2. 802.11 (this has about 20 sub fields for attributes like frame control id, duration, control type, source MAC address, sequence number, service set identifier (SSID), etc.)
3. Encrypted Data (this has sub fields for Key number, frame body).
4. Raw Data (in coded or encrypted form).

Some of the important fields in the wireless packet include: sequence number, used to terminate endpoint and to ensure packets arrive in sequence. The Media Access Control address (MAC) or Ethernet Hardware Address (EHA) or hardware address or adapter address is a quasi-unique identifier attached to most network adapters. It is a number that acts like a name for a particular network adapter, for example, the network cards (or built-in network adapters) in two different computers will have different names, or MAC addresses. Service Set Identifier (SSID) is a code attached to all packets on a wireless network to identify each packet as part of that network. All wireless devices attempting to communicate with each other must share the same SSID.

Types of Wireless Network Attacks

Wireless network attacks belong to five classes
1. Access Control Attacks (e.g., WEP attack, war-driving attacks, Rogue Access points, MAC spoofing): These attacks penetrate a network by using wireless or evading WLAN access control measures like MAC filters.
2. Confidentiality Attacks (e.g., WEP key-cracking, Man-in-the-middle): These attacks intercept private information sent over wireless associations by capturing data frames.
3. Integrity Attacks (e.g., Replay and 802.11 Frame Injection): These attacks send forged, control, management or data frames over wireless networks to mislead the recipient or facilitate another type of attack (e.g., DoS).
4. Authentication Attacks (e.g., Application Login Theft): Attacker captures user credentials (e.g., e-mail address and password) from cleartext application protocols like WinSniffer (http://www.winsniffer.com/) installed by attacker through back door.
5. Availability Attacks (e.g., RF Jamming): These attacks impede delivery of wireless services to legitimate users, either by denying them access to WLAN resources or by crippling those resources.

1.2 Related Work on Data Mining Based Network IDS

A data mining approach to network intrusion detection provides an opportunity to learn the behaviors of network users by mining the data trails of their activities. While recent research e.g., Clustering [12], MADAM ID [7], ADAM [3], MINDS [5], [7] have investigated data mining for intrusion detection, considerable challenges remain unexplored. This involves intrusion detection models for wireless networks not requiring hard-to-get training data in wired network environment, as well as intrusion detection that has no prior knowledge of relationships between attack types and attributes of the network audit data [4]. One of the most recent wired IDS by Zhong et al. [12] applied multiple centroid-based unsupervised Online K-Means clustering algorithm for intrusion detection, with a simple yet effective self-labeling heuristic for detecting attack and normal clusters of network traffic audit data. Some of the drawbacks of this Zhong et al. work are: they used only metrics available in the recorded wireless logs rather than all that are theoretically required to model common wireless attacks. Our work addresses this constraint. Secondly, the assumption that the largest cluster is attack free does not always strictly hold true during DOS attacks, and the authors acknowledge it. Thirdly, this clustering-based intrusion detection cannot detect intrusions or anomalies in real time and therefore not suitable in a sensor network. Other modified systems similar to Zhong et al. that suffer from some of these drawbacks include system by [9]. Applying a density-based outlier de-
tection algorithm to the problem of intrusion detection has proved to be beneficial and has been explored in parts of such systems as MINDs [5]. Some of MINDs' connections are fed to the anomaly detection modules that use a local outlier approach (LOF) detection algorithm [6] to assign a score that reflects how anomalous the connection is, compared to the normal network traffic. However, MINDs analyzes data from one source point (CISCO router), and this is not suitable for wireless networks where attacks come from different directions. MINDs algorithm is not designed for streaming data. MINDs cannot scale to very large network traffic data sets due to the streaming nature of network data. LOF depended on training data. MINDs mode of operation is offline.

**LOF and LSC Outlier Detection Algorithms**

The two outlier detection algorithms mostly related to this work are the density-based clustering algorithms, LOF [6] (local outlier factor) and LSC (local sparsity coefficient) [1] algorithms. LOF is a density based outlier detection algorithm, which assigns each data object in the data set, a degree of being outlier. This degree is called the local outlier factor (LOF) of a data object. LOF is able to capture both outliers ($p_1$ an outlying object at a distance close to a very dense but small cluster, but shorter than the neighboring distances of a second non-dense but large cluster) and ($p_2$ an outlying object far from the dense cluster) due to the fact that it considers the density around the points not basically the simple distance. Based on LOF, a degree of being an outlier is assigned to each data object (connection records). The LOF of a specific data sample $K$ represents the average of the ratios of the density of the example $K$ and the density of its neighbors. LOF requires the neighborhood around all data points be constructed and this involves calculating pairwise distances between all data points, which is an $O(n^2)$ process. This makes it computationally infeasible for millions of data points. To address this problem, systems like MINDs, based on LOF algorithm, sample a training set from the data and compare all data points of this small set. As training data are generally unavailable, we choose to use local sparsity coefficient outlier detection algorithm (LSC-Mine) [1] since LSC algorithm is a density based outlier technique similar to LOF that eliminates some of these drawbacks of the LOF algorithm. LSC algorithm detects outliers based on the distances of objects from their nearest neighbors without actually computing their reachability distances and local reachability densities as done by LOF. This addresses the problem of huge repetitive computation and comparison for every object before the few outliers are detected. Expensive computations might make scalability of these techniques to important applications, like quick intrusion detection infeasible. Given the data objects, and an integer $K$ representing the number of neighbors each object can have, the local sparsity coefficient, LSC algorithm outputs a ranked list of $n$ objects with highest LSC scores. The higher the LSC score of an object, the more outlying the object is. The LSC-Mine algorithm goes through the following seven steps. (1) Compute the k-distance of each object, (2) Find k-distance neighborhood of each object, (3) Compute local sparsity ratio of each object, (4) Compute the pruning factor of each object, (5) Obtain the candidate set, (6) Compute LSC using the candidate set, (7) Rank outliers as those with the highest local sparsity coefficients. A high local sparsity coefficient indicates the neighborhood around an object is not crowded and hence a higher potential of the object being an outlier whereas a low local sparsity coefficient indicates a crowded neighborhood and hence a lower outlying potential for the object.

**1.3 Contributions**

This paper proposes a Wireless Network Intrusion Detection System (WIDS), which detects MAC spoofing attack (the brain behind 90% of wireless intrusions), by tracking packet sequence numbers. WIDS detects attacks with a few connections (e.g., one to twenty) that may be confidential and integrity attacks and is able to reduce false positive results. WIDS system scientific solution approach, entails applying a hybrid of a density-based and distance based outlier detection clustering algorithm to wireless network connection records promptly captured with a set of proprietary Network Chemistry hardware sensors. Connection records with low anomaly scores (e.g., bottom 30% of ranked anomaly scored records) represent normal clusters while those with high anomaly scores represent intrusions. One of the limitations of existing IDSs addressed by the proposed work is that the existing IDSs build network profiles of clients and only flag alerts when there are significant deviations, causing attacks with few connections (e.g., one to twenty) to pass through undetected.

**1.4 Outline of the Paper**

Section 2 presents the proposed Wireless Intrusion Detection System, WIDS. Section 3 presents the experimental evaluation of our prototype system and section 4 presents conclusions and future work.

**2. THE PROPOSED WIRELESS INTRUSION DETECTION SYSTEM**

Detection of distributed attacks is still a problem for present day IDS and most IDSs are only able to detect PROBE and DOS attacks (and a few other attacks that make use of large number of connections). Attacks that take place using a few (one to twenty) connections are hard to detect by present day IDS (ADAM, MINDS). Having explored all the different wireless attacks and the vulnerabilities, this paper proposes a wireless intrusion detection system (WIDS) to detect these attacks (WEP key cracking, DOS, War driving, MAC Spoofing) and cover the vulnerabilities. WIDS is designed to run in an environment where millions of packets are generated basically every minute.

**2.1 Components of the Proposed WIDS Intrusion Detection System**

The proposed wireless network intrusion detection system, WIDS, consists of a wireless Access Point (AP), one or more Network Chemistry RF Sensors and an Oracle database as shown in Figure 1. Clients connect to the wireless networks through the access point. The Radio Frequency (RF) sensors [8] are used to capture network packets that clients send to the access point. Sensors are purpose-built devices that provide continuous RF surveillance and data aggregation to manage the security of the network. Sensors receive and analyze 802.11 packets, analyze the data, and send processed data to the server, where the information is further analyzed and
the sensors need to be configured so that they can detect
settings.
the client RFprotectClient software choosing all default set-
and choosing the default Firebird database, before installing
installing the sensor server software, RFprotectServerSetup
are (1) Packetyzer, which is an open-source packet capture
Chemistry sensor client software, called RFprotectClientSetup-
GHZ CPU, 1 GB RAM, 60 GB of Hard disk, the Network
personal computer with Windows XP Operating System, 3.0
software (Firebird). Our sensor client is on a separate per-
RFprotectServerSetup-5-0-6-5.exe, which includes the database
Stored wireless packets are captured with sensor in our
Figure 1: The Proposed WIDS System Architecture
stored. For the Sensors to perform their function, we in-
stalled and configured RFprotect Server and Client soft-
ware. The RFprotect Server analyzes, stores, and integrates
data from Sensors. The Server comprises the RFprotect En-
gine, a database of known stations, experts, location anal-
ysis, 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. We use the Console for customizing alert noti-
fication, configuring authorized stations and Sensors; and
shielding against rogue associations. Multiple consoles can
be connected to the Server.
Hardware/Software for Sensor Installation and Con-
figuration
There is need for a server and a client machine although
both server and client software for the sensor installation
can reside on the same machine. Our set up involves a sen-
 sor server on a personal computer with Windows 2003 Server
Operating System, 1 GHZ dual processor, 1 GB RAM, Com-
paq Disk Array SCSI DISC used to prevent loss of data, the
Network Chemistry sensor server software called
RFprotectServerSetup-5-0-6-5.exe, which includes the database
software (Firebird). Our sensor client is on a separate per-
sonal computer with Windows XP Operating System, 3.0
GHZ CPU, 1 GB RAM, 60 GB of Hard disk, the Network
Chemistry sensor client software, called RFprotectClientSetup-
5-0-6-5.exe, which is used to visualize the data captured by
the sensor. Two other needed software on the sensor client
are (1) Packetyzer, which is an open-source packet capture
software, and (2) CommView for WiFi [10], which is a wire-
less packet capture software as well as a sniffer.
The sensor server and client set-up process involves first
installing the sensor server software, RFprotectServerSetup
and choosing the default Firebird database, before installing
the client RFprotectClient software choosing all default set-
tings.
Once the sensor server and client software are installed,
the sensors need to be configured so that they can detect
all wireless networks around them and send data to the sen-
sor server Firebird database and to configure the sensor, we
launch the RFprotectClient, and choose the server to con-
nect to (e.g., localhost), type the username (e.g., SYSDBA),
password (e.g., masterkey) once connected to a server, then
we select the RFprotect Console. Then, we select a sensor to
add, configure its parameters and a server address for the
sensor so that the sensor can communicate with the server.
Monitoring and Capturing Packets
Once the sensors are configured, they are able to detect all
wireless networks around them. Packetyzer is a packet cap-
ture program that is installed with the Network Chemistry
RFprotect software. Thus, Packetyzer has been used to cap-
ture packets in Woddlab. 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 signatures of those attacks are
updated in the RFprotect server.
Sample wireless packets are captured with sensor in our
Woddlab lab showing captured beacon frames. A beacon
frame is a packet sent by a wireless access point (on a reg-
ular basis) in a wireless infrastructure mode operation to
allow wireless clients within the vicinity to detect the Sta-
tion Set Identifier (SSID) of the wireless networks. SSID
defines the name of the wireless network that all the wireless
clients associate with. The first address field in the beacon
frame is the destination address. This field has a value of
“ff:ff:ff:ff:ff”, this value indicates that the packet is to be
sent to all stations. The third address field in the BSSID
(Basic Station System ID) field which contains the MAC ad-
dress for the wireless side of the access point. Another field
to note in this packet is the sequence number field, which
is incremented by one every time the wireless station emits
a packet. Thus, since the sequence number for this beacon
frame packet is 4025, the next packet that will be emitted
by this wireless station would then be 4026. Promptly an-
alyzing these attributes of captured wireless packets with
our data mining (clustering) based proposed technique will
detect wireless intrusions.

Pre-processing Sensed Packets from Sensor Database
One of the core contributions of this paper is that of select-
ing features that can effectively detect wireless attacks. At-
tackers 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 T time interval, play vital roles
in detecting attacks. We use CommView for WIFI (wireless
fidelity), a powerful wireless network monitor and analyzer
for 802.11 a/b/g/n networks to pre-process captured wire-
less sensor logs in WLAN traffic.
With CommView for WIFI, sensed packets are exported
into a .CSV file. The data in .CSV format are then cleaned
and processed by converting missing values to zero. We also
removed features like IP address, since wireless clients roam
about continuously, their IP addresses change as they roam
from one AP to another. We use MAC addresses instead of
IP address to track records of each client. With SQL-
Loader utility in Oracle, the data are loaded into an Oracle
database using the appropriate script file entries.
From CommView for WIFI statistics report of network ac-
tivities, we derived connection records like total number of packets, packet bytes sent, and number of RTS retries, which help us in detecting attacks with few (one or twenty) connections, as well as DOS attacks. Features extracted from the sensor log include Time, Src/Dest IP, Packet size, BSSID, Frame type/ Subtype, Rate, Client AP sequence number, Signal, ESSID, Source type, Channel.

2.2 The Proposed WIDCA Algorithm

The main algorithm based on the LSC outlier detection algorithm, for clustering and identifying outlying or abnormal connection wireless records, which is used by the proposed WIDS system as shown in Figure 1 is Wireless Intrusion Detection Clustering Algorithm (WIDCA), shown formally as Algorithm 1. Given the wireless record-set, a set of records containing wireless data with parameters (mac address, no of packets sent, no of packets received, packet byte, channel, signal, rate, errors, retries, packet size, receiving time), the WIDCA algorithm outputs spoofed address, rogue APs, alert clusters with anomaly records by going through the following three steps.

Step 1: Calculates all MAC addresses with large gaps in sequence number as spoofed and any change in MAC/channel pair AP as rogue and this it does by calling an algorithm for performing sequence number capture (see Sequence-number-capture Algorithm 2).

Step 2: Using the LSC-Mine outlier detection algorithm, compute outlying connections as the top strongly ranked outliers and use them as clustering centroids (LSC-Mine details provided in the related work section).

Step 3: Using our Online Wireless clustering algorithm, identify anomalous records as those that have their distances from their closest cluster more than the minimum deviation distance, D as anomaly (see Wirelesscluster Algorithm 3). D is the maximum of the distances between pairs of cluster centroids.

The formal presentation of the main WIDCA algorithm is given as Algorithm 1.

ALGORITHM 1. (WIDCA: Detecting Anomalous Wireless Packets)

Algorithm WIDCA()
Input: records, $r$, a set of wireless records with parameters (mac address, no of packets sent, no of packets received, packet byte channel, signal, rate, errors, retries, packet size, receiving time)
Output: spoofed address, rogue APs, alert clusters as anomaly records
begin
  // find spoofed MAC addresses as those with large gaps in sequence number and change in MAC/channel pair AP
  as rogue, see algorithm 2
  (1) Spoofed MAC/AP addresses = Sequence-number-capture(records);
  (2) Initial cluster centroids = LSC-Mine(record-set)
  // 30% bottom records with low LSC scores have low anomaly
  // scores and are initial centroids //
  (3) Anomalous Packet = Call wirelesscluster(anomaly records)
  // Records with distance from the closest cluster more
  // than minimum deviation, D, are anomalous.
end

2.3 Application of WIDCA Algorithm on Sample Data

Example 1: Given seven records $M_1$ to $M_7$ representing seven wireless connection records, show how the proposed WIDCA algorithm can be used to cluster these records as either normal or intrusive.

Solution 1: After identifying rogue MAC and AP addresses in step one of Algorithm 1, WIDCA, the records are passed to step 2. The LSC outlier algorithm next assigns anomaly scores to the records and selects the 30% bottom low outlying records or more normal records as initial centroids for running the Wirelesscluster algorithm (a type of online K-means algorithm where the initial centroid is not randomly selected and the minimum deviation distance D is changed each time a record is assigned to a cluster).
When LSC-Mine is called in step 2 of WIDCA, it starts by computing K-distance of the 7 records with K assumed to be 4. K is the minimum number of objects desired to be in a neighborhood of an object O. K-distance of M, is the maximum distance from object M when every object in the data set is considered to have at least K-neighbors. A pre-processing of connection records (5 records shown) to translate attribute values to numeric distance values yields the distance between objects as shown in Table 1. Table values for M₀ and M₇ are respectively M₀(12, 88000, 18, 42, 395693, 0, 167) and M₇(179, 87833, 185, 125, 395, 526, 167, 0). LSC-Mine then, selects the first 4 (since K=4) distinct minimum distances from all objects M₁ to M₇. For example, for M₁, they are: 6, 54, 12, 179. Next, LSC-Mine selects the maximum of the first 4 minimum distances as the K-distance(M₁)=max(6, 54, 12, 179) = 179. The K-distances of the other objects are obtained in a similar fashion. Step 2 of the LSC-Mine algorithm finds K-distance neighborhood of M denoted (Nₓ(M)), which contains every object with distance not greater than K-distance (M). The rationale for computing the K-distance neighborhood is to find the nearest neighbors of each object. For instance, K-distance neighborhood of M₁ contains M₃, M₄, M₆ and M₇ since K-distance (M₁) is 179. Similarly, the Nₓ(M₂...M₇) are obtained. Step 3 of the LSC-Mine computes the local sparsity ratio of an object M denoted as lsrₓ(M), which is the ratio of the cardinality of the result obtained by dividing the number of objects in the neighborhood of M by the sum of all the actual distances in that neighborhood. \( \text{lsr}(M) = |N_k(M)|/(\sum_{i \in N_k(M)} \text{distof}(N_i)) \). The distance of an object Xᵢ is the Euclidean distance of Xᵢ from other data points Yᵢ and is computed as:
\[ \text{distof}(X_i) = \sqrt{\sum (X_i - Y_i)^2} \].

\( \text{lsr}_k(M) = 4/89 = 0.045, \) where 4 is the number of object in the neighborhood of M₁ and 89 is the sum of the M₁ neighborhood distances. The LSC(M) is computed as the average ratio of the local sparsity ratio of M to that of its K-nearest neighbors. \( \text{LSC}(M) = (\text{lsr}_k(M_2) + \text{lsr}(M_3) + \text{lsr}_k(M_6) + \text{lsr}_k(M_7)) / 4. \) The LSC for M₁...M₇ are computed to be respectively: 0.029, 18004.39, 1.38, 1.49, 3409.27, 1.18, 3.73. This leads to an LSC ranking showing the strongest outliers ranked 1st and the weakest ranked 7th as: M₁ being the 1st scored outlier, the 2nd is M₂, the 3rd is M₇, the 4th is M₄, the 5th is M₃, the 6th is M₆ and the 7th is M₇.

Thus, since the WIDCA algorithm uses the bottom 30% of ranked outliers as initial centroids for clustering, M₁, M₃, M₆, and M₇ are used for initial centroids as they represent the most records that are free of intrusions. The minimum deviation distance D is computed with these three records as M = 572.31. For each new connection record, WIDCA selects the closest cluster to place it in if the record value is not more than the minimum deviation, D. If it is more than D, it is moved to the alert cluster. For example, new record M₁₄ with attribute values 265712, 88214, 10226569, 5654318, 289271, 5933, 98, 5.5 will go to the alert cluster. The new record M₁₅ with values 10, 2, 8, 131, 2, 0, 43, 1 is inserted into the closest normal cluster C₁ and the centroid for C₁ is re-computed as: 7 + 0.05(10-7), 3 + 0.05(2-3), 348 + 0.05(8-348), 225 + 0.05(131-225), 7 + 0.05(2-7), 0 + 0.05(0-0), 79 + 0.05(43-79), 1 + 0.05(1-1). New minimum deviation (M) is calculated by computing the distance from new C₁, C₂ and C₃. The process will be repeated for other records. The alert cluster is examined by a human analyst for any new intrusion signatures to be added to the attack database.

### 3. Testing and Implementation Environment

Our prototype WIDS system is developed in Java language under windows platform. The experiment is divided into three parts. We tested our system along with one other system by using crafted wireless intrusions, to see how many intrusions each system can find. In the second part, the efficiency of the two systems is tested by using continuous crafted wireless intrusions and comparing the CPU usage on the computer that is hosting the IDS system. In the third part, we tested our system in a real network by using crafted intrusions to see how it performs in real world. Our testing environment consists of 3 computers as: C1 is running our system (WIDCA) software implementations with hardware Sensor configured as described in Section 3. Snort-Wireless is installed in C2 and C3 is Online K-Means [12]. We have a laptop booted with BackTrack network security suite (http://www.remote-exploit.org/backtrack.html) from where we launch our attacks. BackTrack is a Linux tool distributed as a Live CD and includes over 300 security tools that we used in crafting attacks. All the computers were connected through a wireless router, which ensures that the hardware Sensor captures all traffic including: 1. Confidentiality Attacks, 2. Access Control Attacks, 3. Availability Attacks, 4. Integrity Attacks, and 5. Authentication Attacks.

We measured the accuracy of the detection by using Detection-Rate defined as: Detection Rate = (Number of Intrusions found) / (Total number of Intrusions). Table 2 shows the results of our experiment comparing our WIDCA algorithm with Snort-Wireless with On-line K-Means algorithm.

### Testing for False Positives

In this experiment, we tested to see if there is reduction in

| Table 1: Distances between 5 of 7 Example Records |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|                | M₁    | M₂    | M₃    | M₄    | M₅    |
| M₁    | 0     | 88012 | 6     | 54    | 395705|
| M₂    | 88012 | 0     | 88018 | 87958 | 307693|
| M₃    | 6     | 88018 | 0     | 60    | 395711|
| M₄    | 54    | 87958 | 60    | 0     | 395651|
| M₅    | 395705| 307693| 395711| 395651| 0     |

<table>
<thead>
<tr>
<th>Table 2: Detection Rate Results of 3 Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacks</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Confi(5)</td>
</tr>
<tr>
<td>Access(5)</td>
</tr>
<tr>
<td>Avail(11)</td>
</tr>
<tr>
<td>Integ(5)</td>
</tr>
<tr>
<td>Authen(9)</td>
</tr>
<tr>
<td>Detect Rate(35)</td>
</tr>
</tbody>
</table>
the number of false positives in our system. In false positive, a system flags an alert when in reality there is no attack. To perform this, we crafted normal and attack packets which we launched in our network. The results are shown in Table 3. We have to note here that we increased the rate at which some of these packets were sent so that it looks like a DoS attack when in reality the packets are normal. As can be seen, our system flags few normal packets as attacks compared with the two other systems (Snort and Zhong et. al. 2005 clustering approaches). The nature of the packets may have contributed to the reason the third system (Zhong et. al. 2005 clustering approach) has a high false positive.

<table>
<thead>
<tr>
<th>Packets</th>
<th>WIDCA</th>
<th>Snort-W</th>
<th>Zhong’s</th>
</tr>
</thead>
</table>

6. REFERENCES


5. ACKNOWLEDGMENTS

This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an operating grant (OGF-0194134) and a University of Windsor grant.