

University of Windsor

## Scholarship at UWindor

---

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

---

2008

### WiFi Miner: An online apriori and sensor based wireless network Intrusion Detection System

S S Ahmedur Rahman  
*University of Windsor*

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

---

#### Recommended Citation

Rahman, S S Ahmedur, "WiFi Miner: An online apriori and sensor based wireless network Intrusion Detection System" (2008). *Electronic Theses and Dissertations*. 8006.  
<https://scholar.uwindsor.ca/etd/8006>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email ([scholarship@uwindsor.ca](mailto:scholarship@uwindsor.ca)) or by telephone at 519-253-3000ext. 3208.

# NOTE TO USERS

This reproduction is the best copy available.

**UMI**<sup>®</sup>



# WiFi Miner: An Online Apriori and Sensor Based Wireless Network Intrusion Detection System

By

S S Ahmedur 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

2008

© 2008 Ahmedur Rahman



Library and  
Archives Canada

Bibliothèque et  
Archives Canada

Published Heritage  
Branch

Direction du  
Patrimoine de l'édition

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file* *Votre référence*  
*ISBN: 978-0-494-47078-7*  
*Our file* *Notre référence*  
*ISBN: 978-0-494-47078-7*

**NOTICE:**

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

**AVIS:**

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

  
**Canada**

### **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, WiFi Miner, which applies an infrequent pattern association rule mining Apriori technique to wireless network packets captured through hardware sensors for purposes of real time detection of intrusive or anomalous packets. Contributions of the proposed system includes effectively adapting an efficient data mining association rule technique to important problem of intrusion detection in a wireless network environment using hardware sensors, providing a solution that eliminates the need for hard-to-obtain training data in this environment, providing increased intrusion detection rate and reduction of false alarms.

The proposed system, WiFi Miner solution approach is to find frequent and infrequent patterns on pre-processed wireless connection records using infrequent pattern finding Apriori algorithm also proposed by this thesis. The proposed Online Apriori-Infrequent algorithm improves the join and prune step of the traditional Apriori algorithm with a rule that avoids joining itemsets not likely to produce frequent itemsets as their results, thereby improving efficiency and run times significantly. A positive anomaly score is assigned to each packet (record) for each infrequent pattern found while a negative anomaly score is assigned for each frequent pattern found. So, a record with final positive anomaly score is considered as anomaly based on the presence of more infrequent patterns than frequent patterns found.

*Keywords:* Data mining, wireless network intrusion detection, Apriori, infrequent patterns, training data.

## **Acknowledgement**

I would like to thank all people who have helped, inspired and motivated me to write this thesis. I also would like to express my heartfelt gratitude to my wife and other family members including my parents, brother and sister for their inspiration, support and love throughout my MSc study, without which it would not be possible.

I would like to thank my supervisor Dr. Christie Ezeife for her continuous help and encouragement throughout the period of writing this thesis. Her professional guidance, direction, discussing research methodologies, 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. Kemal Tepe, internal reader Dr. Akshaia Aggarwal, and thesis committee chair, Dr. Xiaobu Yuan for making time to be in my thesis committee, reading the thesis and providing valuable suggestions, which have helped me a lot to improve the quality of this thesis.

Finally, I would like to express my appreciation to all my friends and colleagues in WODDLAB for their help and support.



## Table of Contents

Author's Declaration of Originality.....	III
Abstract.....	IV
Dedication.....	V
Acknowledgement.....	VI
Table of Figures.....	IX
Table of Tables.....	XI
1. INTRODUCTION .....	1
1.1. Thesis Contribution.....	4
1.2. Outline of Thesis.....	5
1.3. Network Intrusion Detection for wired network.....	5
1.3.1. Type of Intrusions:.....	5
1.3.2. User to Root Attack: .....	6
1.3.4. Denial of Service Attack:.....	8
1.3.5. Probes Attack:.....	9
1.4. Network Intrusion Detection for wireless network.....	9
1.4.1. Wireless network classification: .....	9
1.4.2. Infrastructure based wireless network: .....	10
1.4.3. Ad-hoc Network: .....	11
1.4.5. Wireless standard for WLAN (IEEE 802.11):.....	12
1.4.6. Differences between Wired Intrusion Detection and Wireless Intrusion Detection:.....	12
1.4.7. Wireless Intrusion Types: .....	15
(1) Passive Attack: .....	15
(2) Active Attacks:.....	16
(3) Man-in-the middle attacks:.....	17
(4) Jamming attacks: .....	18
1.4.8. Counter Measures to wireless security threats.....	18
2. RELATED WORKS.....	21
2.1. Data Mining Approaches for NIDS: .....	21
2.1.1. Association Rule:.....	21
2.1.2. Classification Rule:.....	22
2.1.3. Clustering:.....	25
2.2. Data Mining and Network Intrusion Detection System.....	29
2.2.1. Data Mining Based NIDS:.....	30
2.2.2. Source of Audit Data: .....	31
2.2.3. Processing of Audit Data and Feature Construction:.....	33
2.2.4. Data Mining for NIDS at Alarm Level:.....	34
2.2.5. Review of Data Mining based NIDS and Why Association Rule Mining: ....	36
2.3. Association Rules Mining:.....	37
2.3.1. Apriori Algorithm .....	37
2.3.2. FP-Growth Algorithm:.....	40

2.3.3.	Frequent Episode Rule:.....	42
2.4.	Data Mining based NIDS Projects using Association Rule Mining .....	43
2.4.1.	ADAM .....	44
2.4.2.	MADAM ID.....	46
2.4.3.	LERAD .....	48
2.4.4.	MINDS.....	51
2.5.	Wireless Intrusion Detection System (WIDS).....	57
2.6.	Data Mining in WIDS .....	59
2.6.1.	A Clustering Approach to Wireless Network Intrusion Detection .....	59
2.6.2.	A Hybrid IDS for Wireless Intrusion Detection using Association Rule Mining 66	
3.	PROPOSED SOLUTION FOR WIRELESS INTRUSION DETECTION.....	70
3.1.	Overview of the proposed System .....	71
3.2.	Input and Preprocessor Module .....	73
3.2.1.	Network Chemistry Sensor Software Installation Process .....	75
3.2.2.	Preprocessing phase using Commview for WiFi.....	82
3.3.	Anomaly Detection Module.....	83
3.3.1.	Definition and Terminology .....	83
3.3.2.	The Proposed Apriori-Infrequent Algorithm .....	85
3.3.3.	Anomaly Score Calculation .....	88
3.4.	Example Application of the Apriori-Infrequent and Anomaly Score.....	90
4.	EXPERIMENTS AND PERFORMANCE ANALYSIS .....	93
4.1.	Experimental Setup.....	93
4.2.	Attacks Used in Experiment .....	94
4.3.	Test Result and Performance Evaluation .....	106
5.	CONCLUSION AND FUTURE WORKS .....	113
	Bibliography: .....	115
	Vita Auctoris.....	125

## Table of Figures

Figure 1: Infrastructure based Wireless LAN .....	10
Figure 2: Ad-hoc Wireless Network .....	11
Figure 3: OSI Model .....	13
Figure 4: Difference between wired and wireless packet in Data Link Layer.....	14
Figure 5: MAC header format and illustration of Frame Control Bytes.....	14
Figure 6: Telnet Records.....	23
Figure 7: Example RIPPER Rules from Telnet Records .....	23
Figure 8: Demonstration of classification rule (Hunt's method).....	25
Figure 9: K-Means algorithm.....	26
Figure 10: Sample dataset for K-Means algorithm.....	27
Figure 11: K-Means Example.....	29
Figure 12: TCP Header Format/ Format of raw audit binary data.....	32
Figure 13: FP-tree .....	41
Figure 14: Frequent Episodes Rule (SYN flood attack).....	43
Figure 15: LERAD Algorithm ([MC03] page: 602).....	49
Figure 16: LERAD algorithm (Part 2).....	51
Figure 17: MINDS System .....	53
Figure 18: Local Outlier Factor (LOF) approach .....	56
Figure 19: Matrix representation of C1 and C2 at 0 <sup>th</sup> iteration.....	62
Figure 20: Matrix records for iteration1 .....	64
Figure 21: System workflow.....	72
Figure 22: Algorithm 1 – WiFi Miner Workflow.....	73
Figure 23: Input Module and Preprocessor Module .....	74
Figure 24: RFProtect Login Window .....	76
Figure 25: Sensor Configuration Window.....	76
Figure 26: Discover Sensor Window.....	77
Figure 27: Sensor IP address Configuration .....	78
Figure 28: Console window with unknown sensor.....	79
Figure 29: Console window with sensor.....	80
Figure 30: Console with access points and clients.....	81
Figure 31: Captured packets using Packetyzer .....	81
Figure 32: Sample preprocessed csv file output by Commview for WiFi.....	82
Figure 33: Algorithm: Apriori-SmartJoin.....	86
Figure 34: Algorithm: Apriori-Infrequent .....	88
Figure 35: Experimental Network Setup .....	94
Figure 36: Sending Fake Authentication packets .....	96
Figure 37: Saving Fake ARP packets .....	96
Figure 38: Gathering enough packet for WiFiMiner AP to decrypt WEP key.....	97
Figure 39: Processing of data to find the WEP key .....	98
Figure 40: WEP key found for WiFiMiner Access Point.....	98
Figure 41: Fake ARP packets captured and preprocessed by WiFiMiner .....	99
Figure 42: Interface for creating TCP Flood packets.....	101
Figure 43: Interface for creating UDP Flood Attack packets .....	102
Figure 44: SYN Flood Attack packets captured by Commview.....	103

Figure 45: UDP Flood Attack packets captured by Commview..... 103  
Figure 46: Sending Deauthentication packets to the legitimate AP ..... 105  
Figure 47: Deauthentication packets captured by Commview ..... 106  
Figure 48: Comparison of Apriori-Infrequent with Apriori ..... 107  
Figure 49: Innocent packets collected to test the system along with anomalous packets  
(complete log can be found @  
[http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/innocent\\_log.cap](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/innocent_log.cap)) ..... 108  
Figure 50: Preprocessed innocent packets (A complete log can be found @  
[http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/innocent\\_log.csv](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/innocent_log.csv)) ..... 109  
Figure 51: Comparison of Attacks detected and false alarm rate ..... 110  
Figure 52 Specific attacks comparisons of WiFi Miner with SNORT Wireless and  
ADAM ..... 111

## Table of Tables

Table 1: Sample Transaction data.....	21
Table 2: Training data set for classification rule learning .....	24
Table 3: Sample library transactions.....	38
Table 4: Sample database.....	40
Table 5: Ordered Frequent Itemsets.....	41
Table 6: Frequent patterns from FP-tree mining.....	42
Table 7: Schema of attack free database to be used to build attack free normal profile ..	44
Table 8: Sample of normal profile .....	45
Table 9: Sample Network Connection Records.....	47
Table 10: Example output of intrusion pattern from Table 8 .....	47
Table 11: Sample Training Dataset.....	49
Table 12: Time-window based features.....	54
Table 13: Connection-window based features.....	55
Table 14: Metrics used to cluster the wireless logs .....	60
Table 15: Sample Input data for clustering.....	61
Table 16: Sample connection records for kmo algorithm.....	61
Table 17: Object clustering generating cluster ( $C_0$ ).....	63
Table 18: Object clustering generating cluster ( $C_1$ ).....	65
Table 19: Example of data collected over statistical feature set.....	67
Table 20: Example of sample data collected over cross-layer feature set.....	67
Table 21: Market basket format of Table 19 .....	68
Table 22: Market basket format of Table 20 .....	68
Table 23: Selected attributes/ features list for preprocessing .....	83
Table 24: Database records Anomaly Score .....	91
Table 25: Attacks Detected and False Alarm Comparison .....	110
Table 26: Specific Attacks Detection Comparison .....	111

# 1. INTRODUCTION

Security of computer networks has become the most crucial issue nowadays. Traditionally, we consider the firewall as the first line of defense, but the unsophisticated firewall policy cannot meet the requirements of some organizations, which need high security [XQJ01]. Many studies and research have been done already in this field and still this field is getting a lot of attention from researchers and professionals. There are many methods and models like ADAM [BCJ+01], MADAMID [LS00], MINDS [ELK+04], DHP [LXY03], LERAD [MC03], ENTROPY [Yo03] already present today to resolve this problem, but all of them are based on wired network environment. In the last few years, wireless technology has advanced rapidly in user convenience and flexibility but few studies have been done on intrusion detection of wireless network. This is why security of wireless network or detecting intrusion on wireless network has become an important issue among researchers today.

There are two kinds of traditional IDSs [BK03]: 1) misuse detection model and 2) anomaly detection model. Misuse IDS, which uses well-known attack patterns to detect intrusion, for example, models [De87] [SGF+02] [SZ02] [KTK02] which are efficient at catching previously known intrusions, but unable to detect new intrusions based on previously known intrusions. Anomaly IDS, on the other hand can analyze previous intrusion patterns and based on previous patterns it can detect new anomalous patterns. That is why nowadays "*Data Mining*" has become a vital part in network intrusion detection. The intrusion detection model is an old concept, which was first introduced by Denning in [De87]. But data mining is a newer concept in the network intrusion detection model. Research in this area started as early as 1998/ 1999, [LSM99] [LS98] [LP99]. Many studies have been done on how data mining concept can be used efficiently to improve the performance of network intrusion detection model [Le02][Yo03][MC03] [NC03] [BCL02] [BK03] [LS98] [LSC+01a] [LSC+01b]. The newest concern in this field is how to implement data mining based IDS for wireless network. To understand this more clearly, we need to know what is "*Data Mining*", what is "*wireless network*"

and what is “*Network Intrusion Detection*” separately then we can concentrate on how they can be merged together.

Data Mining is knowledge discovery in databases. Data mining techniques help us to discover hidden patterns in database automatically. As a definition, we can say that Data mining is the *automated* process of extracting *hidden predictive* information from a large database. In other words, data mining is *automated statistical analysis* [Th05].

As we know Internet is the network of networks and nowadays almost all the local networks are somehow connected to the outside networks or Internet. Currently, with the growing size of the World Wide Web, the network is also becoming more complex and new intrusions or attacks are coming out every day. To detect an intrusion all the systems use sensors in form of either hardware or software to monitor the network traffic and raise alarms when they match saved patterns [CG00]. Security analysts decide whether it is a false alarm or a true alarm, then respond accordingly.

Wireless networks are commonly implemented because of the ease of deployment and their ability to provide network access to areas where running cable is not an option. Wireless networks allow employees to roam offices and buildings and provide guests with internet access. However, this same ease of access and mobility can also be leveraged by malicious individuals, who launch attack from the most unlikely of locations. Wireless networks do not have defined borders and air waves can penetrate unintended areas allowing attackers to bypass perimeter firewalls, sniff sensitive information, access internal network or attack wireless hosts without direct access to the network. Proper design of a wireless network can help minimize wireless threats, but like wired networks, defense in depth should be implemented to minimize risk [DH06].

If the network is small and signatures are kept up to date, then an analyst can observe all alarms and can determine the type of attack, if it is a new or old type of attack. But as the network grows and becomes more complex, human analyst will be overwhelmed with all

alarms produced by the system daily. Data Mining can help automate the process so that it can detect new intrusions without the too much help from a human analyst.

Data mining can be very helpful in finding intrusions in networks. For this purpose, all inbound and outbound network traffics are kept in a database and interesting or informative parameters like source IP, destination IP, source MAC address, destination MAC address, timestamps etc are extracted from the database during preprocessing phase. Then, these preprocessed data act as input to data mining model and various data mining techniques like association rule, clustering rule, classification rule are applied to find hidden relationships or rules among these parameters. These new rules are applied to detect any substantial deviation of incoming network traffic to flag it as an anomaly or new type of attack. Data mining has been proven to be efficient in wired intrusion detection system in various works done by many researchers, which include ADAM [BCJ+01], MADAMID [LS00], MINDS [ELK+04], DHP [LXY03], LERAD [MC03] etc. Two major limitations of these data mining based network intrusion detection systems are high rate of false alarm and these systems can be deployed only over wired network. There are two groups of researchers focused on reducing the rate of false alarms using data filters. One group [BCH+01] [BTS+01] used data filters before data are sent to the classifier to be trained. Another group [CG00] used data filters on the output (alarms) of an existing intrusion detection system. The second limitation is more concerning since almost all devices support wireless communication and networks are going from wired to wireless mode. There are many commercial products like AirMagnet [Air08a], AirDefense [Air08b] which offer the functionality of detecting wireless attacks but all are equivalent to misuse intrusion detection systems of wired NIDS, which use “signatures” to detect attacks whose behaviors are well understood. Very recently, data mining has been taken into consideration to enhance the power of intrusion detection capability for wireless networks. We can see that clustering technique has been adopted in [ZKN05] and association technique has been adopted in [LLM+07] [ZZW08] to detect intrusions in wireless networks, which are data mining based Wireless Intrusion Detection Systems (WIDS).



This thesis work proposes a network intrusion detection system for wireless environment using wireless sensor to capture wireless traffic and an online apriori based data-mining algorithm to detect new attacks. Our proposed algorithm (the Real-time Online Apriori algorithm), which has introduced the technique for the first time to analyze the incoming dataset and find infrequent patterns without any prior training with safe data, can detect new types of wireless attacks efficiently with a reduced complexity in comparison to traditional apriori based systems and can flag anomalous connections in real time on the fly.

### **1.1. Thesis Contribution**

This thesis proposes a wireless intrusion detection system called WiFi Miner, with the following objectives:

1. Eliminating the need for hard-to-get training data. This it does with a proposed Online Apriori-Infrequent algorithm, which does not use the confidence value parameter and does not create any rules, but efficiently uses only frequent and non-frequent patterns in a record to compute an anomaly score for the record to determine whether this record is anomalous or not on the fly.
2. Real-Time Detection of Intrusions: This, our system does by integrating proprietary hardware sensors, where streams of wireless packets (e.g., MAC frames) from Access Points (AP) are promptly captured and processed with the proposed Online Apriori-Infrequent algorithm.
3. Our proposed Real-time Online Apriori-Infrequent algorithm improves the join and prune steps of the traditional Apriori algorithm, detects frequent and infrequent patterns in connection records and increase the efficiency and run times significantly. The proposed system targets mostly active wireless attacks, which are not easily detected by existing wired IDSs.

## **1.2. Outline of Thesis**

The thesis is organized as follows: the rest of chapter 1 introduces network intrusion detection system concept for both wired and wireless environments, differences between wired and wireless intrusion and classification of wireless attacks. Chapter 2 discusses how data mining can be used in the field of intrusion detection, major data mining approaches for NIDS, current existing network intrusion detection systems for wired network and wireless 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 and finally chapter 5 presents conclusions and future directions.

## **1.3. Network Intrusion Detection for wired network**

**Definition:** Network Intrusion detection System (NIDS) is a type of security management system for networks. An NIDS collects and analyzes information from various areas within a network to identify possible security breaches, which include both intrusions or attacks from outside the network and from within the network. According to [HLM+ 90], network intrusions are, "Any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource". It does not include local intrusions like file-system virus, local brute force password testing etc [EDA08].

### **1.3.1. Type of Intrusions:**

According to [EDA08], primarily there are four kinds of intrusions:

- User to Root Attack (U2R)
- Remote to User Attack (R2U)
- Denial of Service Attack (DoS)
- 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 [EDA08].

### **1.3.2. 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. Intruder exploits some software vulnerabilities to gain root access. Examples of U2R attack are Eject and Fbconfig.

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.

The eject attack exploits a buffer overflow vulnerability in eject program. "Eject" is a utility distributed in Sun Solaris 2.5. This "eject" utility is used by the removable media devices that do not have an eject button or that are managed by the Volume Management. Due to an insufficient bound checking of the arguments in the volume management library, it is possible to overwrite the internal stack space of "eject". If it is exploited, this vulnerability can be used to gain root access [RB03]. According to [RB03], "The Eject attack consists of 4 steps: i) inject the exploit script to the victim's host computer; ii) compile the exploit script; iii) execute the compiled exploit script; and iv) use the root console. If the exploit script is already in the victim's host and if it has been compiled, then the first two steps become unnecessary."

The attack traces consist of hundreds or thousands of lines of system calls. However, there are only a couple of system call sequences that are sufficient to completely define the attack. Another identifying string that characterizes the eject exploit is `usr/bin/eject` or existence of string `./ejectexploit` or `./eject` [RB03].

### **1.3.3. 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 [Linc07], in this type of attack, the attacker sends 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. The simulation of the attack is described below.

The implementation consists of a carefully constructed mail message which, when sent to the victim machine with a vulnerable version of sendmail, adds a new entry with root privilege to the end of the password file on the victim system. Once this new entry has been added, the attacker can log into the machine as this new user and execute commands as a root user. Initially, the attacker sends a carefully crafted e-mail message to the victim machine. After that the sendmail daemon starts to process this message, overflows one of its buffers, and executes the attacker's inserted commands that create a new entry in the password file. Then, the attacker comes back to the victim machine and uses the new

password file entry to gain root access to the victim machine and perform some malicious actions.

#### **1.3.4. Denial of Service Attack:**

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.

According to [Linc07], A SYN Flood is a denial of service attack to which every TCP/IP implementation is vulnerable (to some degree). Each half-open TCP connection made to a machine causes the 'tcpd' server to add a record to the data structure that stores information describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections. The half-open connections data structure on the victim server system will eventually fill the buffer to hold new incoming connections and the system will be unable to accept any new incoming connections until the table is emptied out. Normally there is a timeout associated with a pending connection, so the half-open connections will eventually expire and the victim server system will recover. However, the attacking system can simply continue sending IP-spoofed packets requesting new connections faster than the victim system can expire the pending connections. As a point of reference, sending 20 SYN packets to a port on a Solaris 2.6 system will cause that port to drop incoming requests for approximately ten minutes [Linc07].

### **1.3.5. 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 [EDA08].

An Ipsweep attack is a surveillance sweep to determine which hosts are listening on a network. This information is useful to an attacker in planning attacks and searching for vulnerable machines. 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 [Linc07]. Then, the attacker can determine which ports on which machines are open and then he can plan to launch other attacks through that open port.

## **1.4. Network Intrusion Detection for wireless network**

The primary purpose of a Wireless Intrusion Detection System (WIDS) is to detect unauthorized wireless access to local area networks and other information assets. According to [Wiki08], these systems are typically implemented as an overlay to an existing Wireless LAN infrastructure, although they may be deployed standalone to enforce no-wireless policies within an organization.

### **1.4.1. Wireless network classification:**

According to the network formation and architecture wireless network can be broadly classified into two categories:

- Infrastructure based wireless network
- Ad-hoc network

### 1.4.2. Infrastructure based wireless network:

In infrastructure based network, there are fixed Wireless Access Points (WAP) and a set of client devices. WAPs are usually connected to a wired network and relay data between client devices on each side. Nowadays, most of the connections among wireless devices occur over infrastructure based service provider like laptops connected to the internet via WAPs. A popular example of this infrastructure based wireless network is WLAN or Wireless Local Area Network.

#### WLAN (Wireless LAN):

With this WLAN users can establish wireless communication within a local area, typically within 100 meters [LLM+07]. WLAN can operate in two modes: Infrastructure based mode and independent (Ad-hoc) mode. In an infrastructure WLAN wireless stations are connected to a wired network such as Ethernet via WAPs. In this network, wireless devices can move within the range of WAP without any disruption to the connection. In independent WLAN, wireless stations within a limited area form a temporary network without using WAPs. This concept is similar to Ad-hoc network, which is described in the next sub-section.

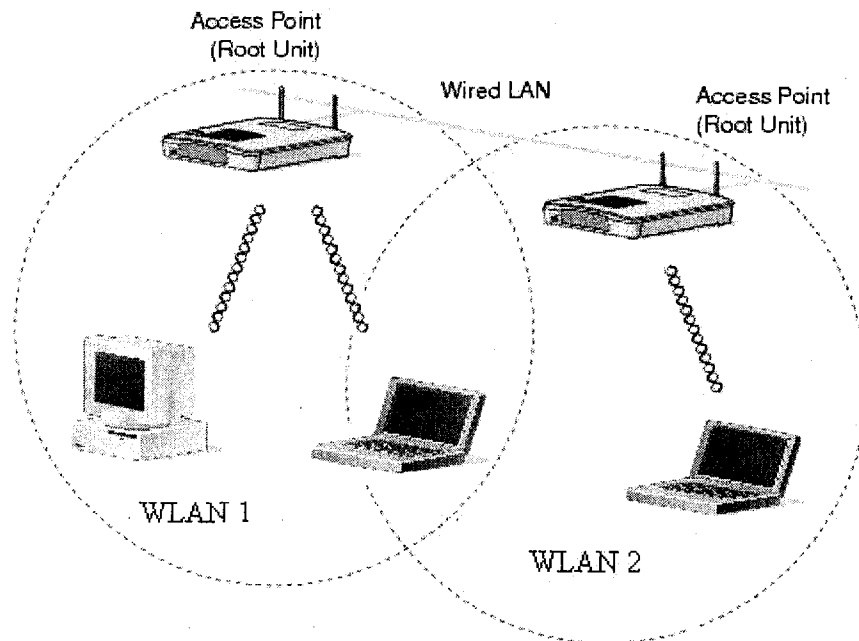


Figure 1: Infrastructure based Wireless LAN

### 1.4.3. Ad-hoc Network:

According to [Mob07], an ad-hoc (or "spontaneous") network is a local area network or other small network, especially one with wireless connections, in which some of the network devices are part of the network only for the duration of a communications session or, in the case of mobile or portable devices, while in some close proximity to the rest of the network. In Latin, *ad hoc* literally means "for this," further meaning "for this purpose only," and thus usually temporary. Ad-hoc networks can perform as stand-alone networks meeting direct communication needs of their users.

For example, Ad-hoc network allows people to come to a conference room and, using infrared transmission or radio frequency (RF) wireless signals, join their notebook computers with other people in the conference to a local network with shared data and printing resources. Each user has a unique network address that is immediately recognized as part of the network. The technology would also include remote users and hybrid wireless/wire connections. In this case, the duration of this temporary network would be the duration of the meeting.

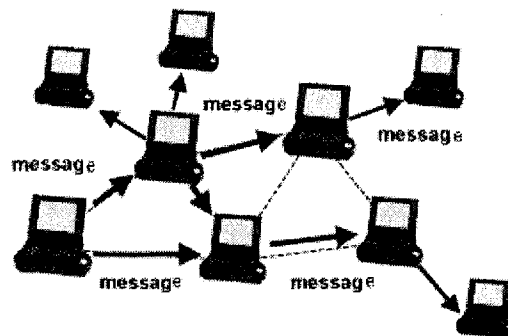


Figure 2: Ad-hoc Wireless Network



#### **1.4.5. Wireless standard for WLAN (IEEE 802.11):**

IEEE 802.11, commonly known by Wi-Fi, denotes a set of Wireless LAN standards developed by working group 11 of the IEEE LAN/MAN Standards Committee (IEEE 802) in 1997. These standards describe everything about Wi-Fi, such as components of the wireless architecture, logical service interfaces, overview of the services, differences between wired LAN and wireless LANs, MAC service definitions, frame formats, authentication and privacy, layer management etc [IEEE 802.11, 1999]. The term 802.11x is also used to denote the set of amendments to the standard. 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.

The built-in security features of 802.11 are provided largely by the Wired Equivalent Privacy (WEP) protocol. In wireless LANs, clients need to be connected with Wireless Access Points (WAPs) in order to communicate with other clients. WEP only provides the security that only authorized clients can be connected to the WAP by providing a correct password phase or a key. Because 802.11b/g has been so widely adopted, the security weaknesses related to WEP and the standard have been exposed and now it is easily breakable [LLM+07].

#### **1.4.6. Differences between Wired Intrusion Detection and Wireless Intrusion Detection:**

The main difference between wired and wireless intrusion detection is that in wired environment the data are transmitted through wire or cable and to detect or intrude into the network, attacker needs to have physical connection over a cable to the network. But in wireless network data are transmitted over air and Wireless networks do not have defined borders and air waves can penetrate into unintended areas allowing attackers to

bypass perimeter firewalls, sniff sensitive information, access the internal network or attack wireless hosts without direct access to the network.

Before knowing the details of wireless and wired IDS, let us know how data is transmitted over network and what OSI (Open System Interconnection) model is. Each piece of information transmitted on a wired or wireless network is sent as packets. A packet has several layers; different layers are dedicated to different specific tasks. Formation and functions of packets can be described best with OSI model which has seven layers. The OSI model is as follows:

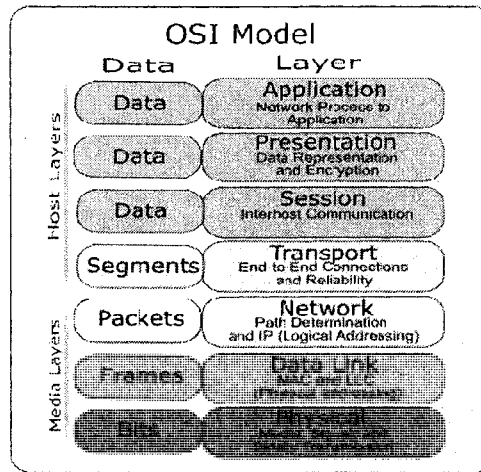


Figure 3: OSI Model

A wireless IDS is unique in that it detects attacks against the 802.11 frame at layer two (Data Link Layer) of the wireless network. This layer 2 (Data Link) is also different in wired and wireless packets. The headers of both packets are shown in figure 4. The upper part of figure 4 shows the wireless (802.11) header format and the lower part of the figure shows the wired (802.3) MAC header format.

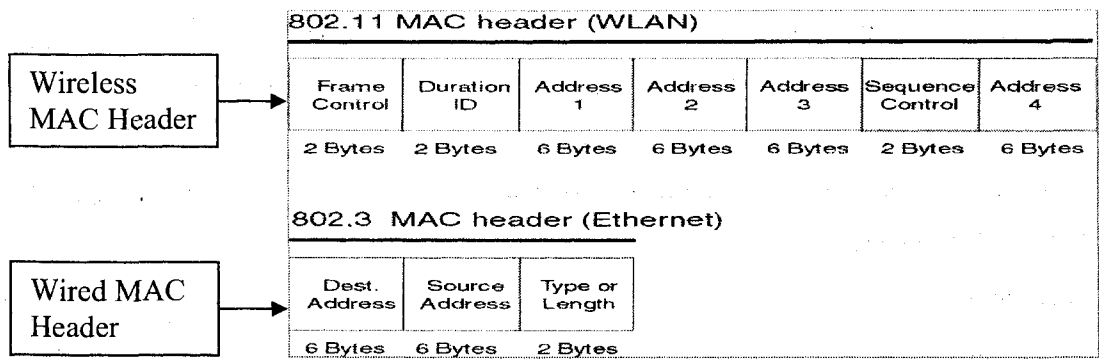


Figure 4: Difference between wired and wireless packet in Data Link Layer

This data link layer contains MAC frames. There are three different types of 802.11 (wireless) MAC frames: data frame, control frame and management frame. Data frames carry protocols and data from higher layers within the frame body. A data frame, for example, could be carrying the HTML code from a Web page (complete with TCP/IP headers) that the user is viewing. Management frames enable stations to establish and maintain communications. The majority of wireless attacks target management frames, because they are responsible for authentication, association, beacons, probe requests/response etc. 802.11 control frames assist in the delivery of data frames between stations. More detailed information on different types of MAC frames can be found at [Wifi07]. Figure 5 shows the general header format of a wireless MAC data frame and the lower part of figure 5 shows the fields in Frame Control in details.

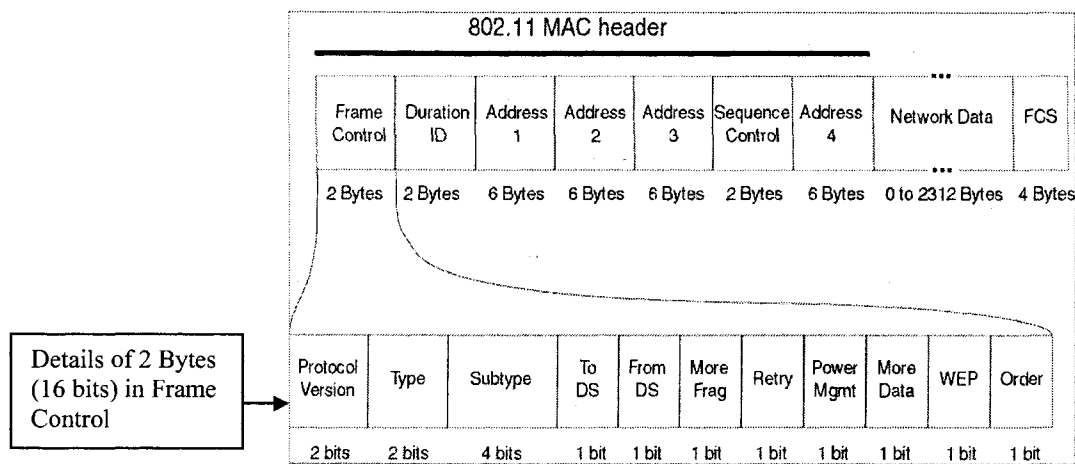


Figure 5: MAC header format and illustration of Frame Control Bytes

Wireless threats like man-in-the middle attacks, rogue access points, war drivers and denial of service attacks function within the 802.11 frames and cannot be detected on layer three (Network Layer) past the access point. Wired IDS will not receive these frames, because management frames are not forwarded to upper layers of the OSI model [DH06].

#### **1.4.7. Wireless Intrusion Types:**

Wireless intrusions can be broadly categorized into four categories [Sh04]:

1. Passive attacks
2. Active attacks
3. Man-in-the middle attacks and
4. Jamming attacks.

Let us review what these attacks mean to the wireless network.

##### **(1) Passive Attack:**

A passive attack occurs when someone listens to or eavesdrops on network traffic [Sh04]. Armed with a wireless network adaptor that supports promiscuous mode, the eavesdropper can capture network traffic for analysis using easily available tools, such as Network Monitor in Microsoft products, or TCPdump in Linux-based products, or AirSnort in Windows or Linux. A passive attack on a wireless network may not be malicious in nature. In fact, many in the war driving (war driving is the act of searching unsecured Wi-Fi networks by a person with a Wi-Fi equipped computer) community claim their war driving activities are harmless or educational in nature. It is worth noting that war driving, looking for and detecting wireless traffic is probably not illegal, even though propagandistic claims to the contrary are often made. Wireless communication takes place on unlicensed public frequencies—anyone can use these frequencies. This makes protecting a wireless network from passive attacks more difficult.

Examples of passive attack are wardriving, eavesdropping etc. Currently, there are many software and tools available for this war driving activity. War driving tools can help an attacker to find and pinpoint available wireless networks. Examples of this kind of tools are NetStumbler for Windows [Sh04], Kismet for Linux [Hu04], KisMac for Macintosh [Hu04] etc. After finding a wireless network, an attacker can do the eavesdropping. In eavesdropping, an attacker simply listens to a set of transmissions to and from different hosts even though the attacker's computer is not taking part to the transaction. Many relate this type of attack to a leak, in which sensitive information could be disclosed to a third party without legitimate users' knowledge. To prevent an eavesdropping attack, one must encrypt the contents of a data transmission at several levels, preferably using SSH, SSL. Otherwise, large amounts of traffic containing private information are passed through air, just waiting for an attacker to listen in and collect the frames for further illegitimate analysis.

## **(2) Active Attacks:**

Once an attacker has gained sufficient information from the passive attack, the hacker can then launch an active attack against the network. There are a potentially large number of active attacks that a hacker can launch against a wireless network. For the most part, these attacks are identical to the kinds of active attacks that are encountered on wired networks. These include, but are not limited to, unauthorized access, MAC spoofing, Denial of Service (DoS) and Flooding attacks.

Once an attacker has found an unsecured wireless network by war driving, he can do the eavesdropping, which is a passive attack and can gain the valid MAC addresses associated with the network. Later he can spoof his MAC address to that authorized MAC address and easily access the network. Even if someone's MAC address is prohibited to a certain network, he can spoof his MAC address to some other MAC address and access the network. Currently, there are many software available in the internet for this MAC spoofing.

Spoofed access points are another problem with the wireless networks, even with WEP authentication. Clients are typically configured to associate with the access point with the strongest signal. An attacker can simply spoof the SSID (the name of the network) of an access point and clients will automatically associate with it and pass frames and messages. Here is where an attacker can capture traffic with time, determine the WEP key used to authenticate and encrypt traffic on the wireless network.

Another example of active attack is “drive-by spamming” which is a variation of drive-by hacking in which the attacker gains access to an unsecured WLAN and uses that access to send huge volumes of spam. Using the drive-by method allows spammers to save themselves the considerable bandwidth costs required to send many messages legitimately, and makes it very difficult for anyone to trace the spam back to its source. A drive-by spamming incident starts with war driving: driving around seeking unsecure networks, using a computer equipped with a wireless Ethernet card and some kind of an antenna. A wireless LAN's range often extends beyond the building housing it, and the network may broadcast identifying information that makes access simple. Once the attacker finds an unprotected e-mail (SMTP) port (port no 25), the attacker can send e-mail as easily as someone inside the building. To the mail server, the messages appear to have come from an authorized network user. In this way a spammer or attacker sends out tens or hundreds of thousands of spam messages using a compromised wireless network.

### **(3) 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 [Sh04] and AiroPeek [Air07b], and physical security.

#### **(4) 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.

#### **1.4.8. Counter Measures to wireless security threats**

To prevent the attacks, wireless networks can adopt a variety of techniques. These techniques can be broadly classified into two categories [LLM+07]:

- Implementing Encryption and Authentication
- Developing IDS solution

In this subsection, we will discuss the current technologies for encryption and authentication. Some of the current IDS solutions for wireless networks will be discussed in Section 2.6.

### **Implementing Encryption and Authentication:**

In 1999, a wireless security encryption standard was introduced as WEP (Wired Equivalent Privacy). This method was introduced as part of the 801.11b standard to provide secure wireless communication using the RC4 stream cipher system from RSA. In cryptography, RC4 (also known as ARC4 or ARCFOUR) is the most widely-used software stream cipher and is used in popular protocols such as Secure Sockets Layer (SSL) (to protect Internet traffic) and WEP (to secure wireless networks). A symmetric encryption scheme is used in WEP, where a shared key is used for both encryption and decryption. It was, however, quickly breached and anyone intercepting and monitoring the wireless traffic could easily break the encryption using a brute force attack with tools such as Aircsnort and WEPCrack [Hu04].

The next technique that was introduced to strengthen the WEP standard was WPA (Wi-Fi Protected Access) in 2003. This technique is supported in the more recent 802.11a and 802.11g networks. It uses Temporal Key Integrity Protocol (TKIP) as an improved approach to key encryption by mixing the keys [Hu04]. According to [Wiki07], "Data is encrypted using the RC4 stream cipher, with a 128-bit key and a 48-bit initialization vector (IV). One major improvement in WPA over WEP is Temporal Key Integrity Protocol (TKIP), which dynamically changes keys as the system is used. When combined with the much larger IV (Initial Vector), this defeats the well-known key recovery attacks on WEP".

Authentication means that only authorized users can access a network. Authentication solutions include the use of usernames and passwords, smart cards, biometrics, PKI or a combination of solutions like smart card with PKI [Li06]. In cryptography, a public key



infrastructure (PKI) is an arrangement that binds public keys with respective user identities by means of a certificate authority (CA).

Security based on cryptography can offer data confidentiality, validity, integrity and authentication. However, some cryptography schemes can be breakable and hence it is not a full proof scheme for intrusion detection. In addition, even if cryptographic designs are not challengeable mathematically, cryptosystems that implement the design may be vulnerable to attack due to software bugs [LLM+07].

## 2. RELATED WORKS

### 2.1. Data Mining Approaches for NIDS:

#### 2.1.1. Association Rule:

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.

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 a 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 [Du03]:

Given a set of items  $I = \{I_1, I_2, \dots, I_m\}$  and a database of transactions  $D = \{t_1, t_2, \dots, t_n\}$  where  $t_i = \{I_{i1}, I_{i2}, \dots, I_{ik}\}$  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.

### **2.1.2. Classification Rule:**

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. For example, consider the telnet connection records shown in Figure 6. Here, *hot* is the count of accesses to system directories, creation and execution of programs, etc, *compromised* is the count of file/path “not found” errors, and “Jump to” instructions, etc. RIPPER (a standard rule based machine learning algorithm developed at ATT research) [LS00], a classification rule learning program, generates rules for classifying the telnet connections and some of the rules are displayed in figure 7.

label	service	flag	hot	failed_logins	compromised	root_shell	su	duration	...
normal	telnet	SF	0	0	0	0	0	10.2	...
normal	telnet	SF	0	0	0	3	1	2.1	...
guess	telnet	SF	0	6	0	0	0	26.2	...
normal	telnet	SF	0	0	0	0	0	126.2	...
overflow	telnet	SF	3	0	2	1	0	92.5	...
normal	telnet	SF	0	0	0	0	0	2.1	...
guess	telnet	SF	0	5	0	0	0	13.9	...
overflow	telnet	SF	3	0	2	1	0	92.5	...
normal	telnet	SF	0	0	0	0	0	1248	...
...	...	...	...	...	...	...	...	...	...

Figure 6: Telnet Records

RIPPER rule	Meaning
guess :- failed_logins >= 5.	If number of failed logins is greater than 5, then this telnet connection is “guess”, a guessing password attack.
overflow :- hot = 3, compromised = 2, root_shell = 1.	If the number of hot indicators is 3, the number of compromised conditions is 2, and a root shell is obtained, then this telnet connection is a buffer overflow attack.
...	...
normal :- true.	If none of the above, then this connection is “normal”.

Figure 7: Example RIPPER Rules from Telnet Records

Here, we see that RIPPER indeed selects unique feature values in identifying intrusions. These rules can be first inspected and edited by security experts, and then incorporated into misuse detection systems.

The accuracy of a classification model depends directly on the set of features provided in the training data. For example, if the features *hot*, *compromised* and *root shell* were removed from the records in figure 7, RIPPER would not be able to produce accurate rules to identify buffer overflow connections. Thus, selecting the right set of system features is a critical step when formulating the decision tree classification tasks [LSM99] [LS98].

Now let's use a generalized practical example.

Outlook	Temp(F)	Humidity(%)	Windy?	Class
Sunny	75	70	true	Play
Sunny	80	90	true	Don't Play
Sunny	85	85	false	Don't Play
Sunny	72	95	false	Don't Play
Sunny	69	70	false	Play
Overcast	72	90	true	Play
Overcast	83	78	false	Play
Overcast	64	65	true	Play
Overcast	81	75	false	Play
Rain	71	80	true	Don't Play
Rain	65	70	true	Don't Play
Rain	75	80	false	Play
Rain	68	80	false	Play
Rain	70	96	false	Play

Table 2: Training data set for classification rule learning.

Suppose we have two classes: Play and Don't Play. We want to observe other attributes and want to conclude our decision among these two classes. At first, we take the class Play. Then we relate it with the first attribute "outlook". "Outlook" can have three values which are "sunny", "overcast" and "rain". From the dataset we can see that if the outlook is overcast then it is always "play", so we can make it as non-expandable leaf, as there is no chance that this condition will be violated. Then if the outlook is "sunny" then we can see that "play" occurs two times and "don't play" occurs three times, so we make "don't play" as expandable leaf. At last when outlook is "rain" we see from the dataset that "don't play" occurs two times and "play" occurs three times, so we make "play" as expandable leaf. Then we look at another attribute "humidity". We can see that when the outlook is sunny, "don't play" occurs 3 times and all those times "humidity" is greater than 75% (90%, 85%, 95%) and two times when "play" occurs "humidity" is less than 75% (70%, 70%). So when the humidity is less than 75% we make "play" as non-

expandable leaf and when “humidity” is greater than 75% we make “don’t play” as non-expandable leaf. This is how we classify each condition into a class until it becomes a non-expandable leaf. The steps and demonstration of this rule is as follows:

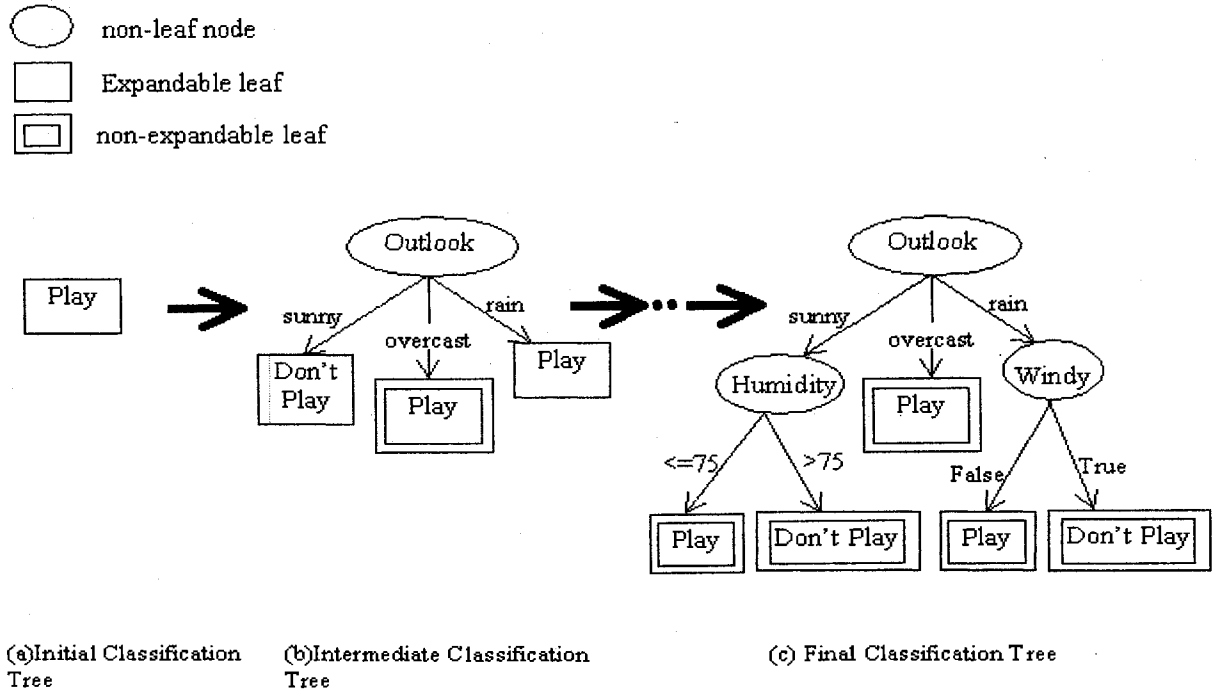


Figure 8: Demonstration of classification rule (Hunt’s method)

### 2.1.3. Clustering:

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 [Be03]. 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 [Be03]. 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 9: K-Means algorithm**

produced by this operation will most probably be a false alarm and can be ignored. To find such sequence they have applied a *Frequent Episode Rules* algorithm, which is a data mining algorithm [CG00]. These algorithms were implemented and tested over wired network intrusion and were not tested for Wi-Fi intrusions. However, the concept of these algorithms can be used for Wi-Fi intrusions, but these algorithms need to be rewritten to be suitable for Wi-Fi intrusions.

### **2.2.5. Review of Data Mining based NIDS and Why Association Rule Mining:**

Various data mining techniques have been applied to intrusion detection because they have the advantage of discovering useful knowledge that describes user's or program's behavior from large audit data sets. Data mining has been applied in two ways in NIDS. One is at TCPDUMP level, which is experimented more frequently and has been developed and improved by many researchers. Second way is at alarm level. Their main goal was to reduce false alarm rate. In normal basic data mining based NIDS we can see that, network traffic data are coming from various sensors. For pre-processing those raw binary data we have used BSM/ BAM/ NFR [LSC+01a]. After preprocessing those data we get those data in a formatted manner with sourceIP, destinationIP, sourcePort etc. Then we have a classifier, where we apply data mining techniques (association rule, classification rule, frequent episode rule, clustering etc.) to train the classifier with huge amount of previously known normal data; so that it can dig out all correlation among the normal datasets and specify them as normal. Then, this trained classifier is placed into real environment and any incoming dataset is compared with those normal datasets in our database. If there is certain deviation we generate a flag. At the same time we keep the training process of classifier on, so that it will be capable of detecting more new attacks.

#### Why Association Rule based NIDS:

In this thesis we have chosen Association Rule based mining for NIDS because it is a straight forward algorithm to implement and very easy to understand. The only limitation



of this approach is scalability: when the dataset becomes huge, the mining process becomes slower as it needs to generate more rules. But we have overcome this problem by introducing an Online Apriori like algorithm, where it uses a faster approach for joining, called *Smart Join* instead of normal Join method, which is the most costly approach in Apriori algorithm. Moreover, our Association Rule based algorithm does not generate frequent rules, which reduces its cost greatly and also eliminates the need for training data. Instead of generating frequent rules and later comparing them with incoming data to find infrequent patterns, it points out the infrequent patterns or anomalies on the fly like clustering methods. The question arises why we did not chose clustering method then. In clustering approaches dealing with large number of dimensions can be problematic because of time complexity and the effectiveness of the method depends on the definition of distance. It is also very critical to determine the cluster borders in clustering approaches. On the other hand for Classification, the accuracy of a classification model depends directly on the set of features provided in the training data and selecting the right set of system features is a critical step when formulating the classification tasks. If the feature set is not chosen correctly, then the result of classification will be totally wrong. So, we have found the Association Rule based mining technique the best and safest one to implement for Network Intrusion Detection environment. In next several sections we will review some important algorithms (Apriori, FP Growth and Frequent Episode) of Association mining technique and some NIDS models based on Association rule mining concept.

## **2.3. Association Rules Mining:**

### **2.3.1. Apriori Algorithm**

The Apriori algorithm was first proposed in [ISA93]. The Apriori algorithm only extracts the frequent patterns from a large dataset with a given support.

Let us discuss an example of how to derive association rules using Apriori algorithm. For example, in a set of book store transactions, the following records were found:

Transaction No	Items Purchased
1	Book, Pencil
2	Book, Pencil, Paper, Magazine
3	Book, Paper, Magazine

**Table 3: Sample library transactions**

A standard algorithm for association rule mining is the Apriori algorithm [ISA93], which works as follows. Suppose a user specifies minSupport is 2. First we will collect the transactional database items and create an itemset Candidate 1 ( $C_1$ ), then we will eliminate all of them from  $C_1$  which do not have support greater than or equal to the minSupport. We will call the second one as frequent patterns with one element in their set,  $L_1$ . After creating  $L_1$  we need to create  $C_2$ . To create  $C_2$  we will join  $L_1$  with itself (Apriori-gen way), and will eliminate all itemsets which do not have at least minSupport. We will continue this process until we get a candidate set  $C_n$  or  $L_n$  empty. After that we will create a large itemset which will be the union of all  $L$  (i.e.  $L=L_1 \cup L_2 \cup \dots$ ). Now let's see the process with sample data. From the sample table given above, we compute the support of each itemset by scanning the database table to find that for example Book:3 meaning that Book has support of 3.

$$C_1 = \{\text{Book:3, Pencil:2, Paper:2, Magazine:2}\}$$

$L_1 = \{\text{Book, Pencil, Paper, Magazine}\}$ , Since all have support  $\geq$  minSupport 2 in the database.

Joining  $L_1$  with  $L_1$  (Apriori join way) gives  $C_2$  and scanning the database table gives the appended supports:

$$C_2 = \{(\text{Book, Pencil}):2, (\text{Book, Paper}):2, (\text{Book, Magazine}):2, \\ (\text{Pencil, Paper}) :1, (\text{Pencil, Magazine}) :1, \\ (\text{Paper, Magazine}) :2\}$$

Support of (Book  $\rightarrow$  Pencil): 2.  $\checkmark$  (OK).

(Book  $\rightarrow$  Paper): 2.  $\checkmark$  (OK)

Support of (Book  $\rightarrow$  Magazine): 2.  $\checkmark$  (OK)

(Pencil  $\rightarrow$  Paper): 1 < 2. X (Need to eliminate)

(Pencil  $\rightarrow$  Magazine): 1 < 2. X (Need to eliminate)

(Paper  $\rightarrow$  Magazine): 2.  $\checkmark$  (OK)

So,  $L_2 = \{(Book, Pencil), (Book, Paper), (Book, Magazine), (Paper, Magazine)\}$

In the same way,

$C_3 = \{(Book, Pencil, Paper):1, (Book, Pencil, Magazine):1, (Book, Paper, Magazine):2\}$

Among itemsets in  $C_3$ , only (Book, Paper, Magazine) has support up to minSupport.

So,  $L_3 = \{(Book, Paper, Magazine)\}$

Then  $C_4 = \{\}$

$L = L_1 \cup L_2 \cup L_3$

= {Book, Pencil, Paper, Magazine,

(Book, Pencil), (Book, Paper), (Book, Magazine), (Paper, Magazine)

(Book, Paper, Magazine)}.

Here L defines all frequent or large pattern from which association rules are generated. From every frequent pattern, e.g., (Book, Paper, Magazine) rules are generated from all of its itemsets and only rules with confidence greater than or equal to the minimum confidence provided by the user are retained, while the rest are pruned. Rules from the frequent pattern (Book, Paper, Magazine) are (Book, Paper)  $\rightarrow$  Magazine, (Book, Magazine)  $\rightarrow$  Paper, (Paper, Magazine)  $\rightarrow$  Book and the confidence of the rule (Paper, Magazine)  $\rightarrow$  Book from the database table is the cardinality of the rule divided by the cardinality of the antecedent equals 2/3 or 75%. Thus, this rule is retained if the minimum confidence is 50%.

Generally, in data-mining based NIDS we create a database of non-intrusive events and then apply association rule technique into that dataset to find out all other rules or events when there will be no intrusions. This will find all hidden normal behavior. Then, these rules will be compared with any incoming data itemsets to determine if it is an intrusion or not. The most critical factor here is that we have to set a minimum threshold for

minimum support and confidence level. Later when discussing the algorithms, we will see it in details.

### 2.3.2. FP-Growth Algorithm:

In [HPY+04], authors have presented a novel method called “FP-Growth”, which mines the complete set of frequent itemset without candidate generation. The FP-growth algorithm transforms the problem of finding long frequent patterns to looking for shorter ones recursively and then concatenating the suffix [HPY+04]. The main strength of this algorithm is that it needs to scan the database only twice to mine the FP-tree and later from this FP-tree frequent patterns can be found easily. The main limitation of this approach is that it needs to build the conditional pattern base and conditional FP-tree recursively, which needs a lot of memory [EDA08].

Let us use the sample database given below in Table 3 to describe the algorithm:

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

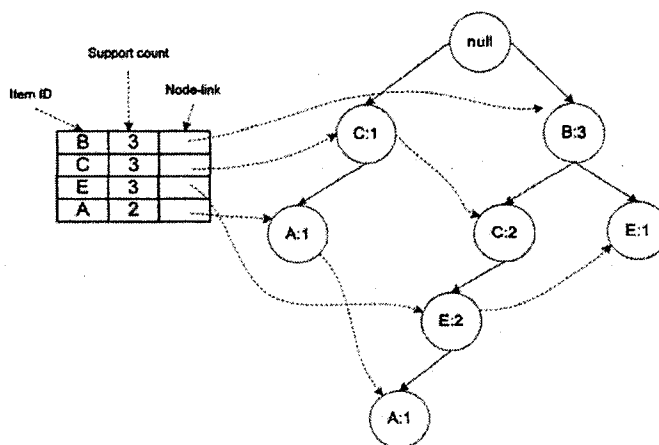
Table 4: Sample database

Step 1: The algorithm will scan the full database and will extract the frequent 1-itemsets and their support counts and sort the list of frequent 1-items in the descending order. Suppose the minimum support is 2. So, item D will be deleted as it has support count 1, which is less than minimum support. So, the ordered list will be:  $L = \{B:3, C:3, E:3, A:2\}$ . So, now the ordered frequent 1-itemsets will be as follows:

TID	Ordered Frequent Items
100	C A
200	B C E
300	B C E A
400	B E

**Table 5: Ordered Frequent Itemsets**

Step 2: Now the algorithm starts constructing FP-tree. First it creates the root of the tree as “null”. Then, it scans the database for the second time and items in each transaction are processed in L order and for each transaction a branch is created. For example, the first transaction “T100: C A” will create the first branch with two nodes. The second transaction “T200: B C E” will construct a different branch with three nodes. The third transaction “T300: B C E A” has a common prefix of “B C E” with T200, so it will follow that branch of T200 and will increase the count of B, C and E to 2 and will add a new node for “A”. The fourth transaction “T400: B E” has a common prefix of “B” with T200, so it will increase the count of “B” to 3 and will add a new node for “E” at a different branch. Moreover, for easily traversing the tree, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links. The FP-tree will look like the following Figure 13:



**Figure 13: FP-tree**

Step 3: Now the algorithm starts to mine the FP-tree. At first consider node “A”, which is the last node in the ordered list “L” and in the tree. Consider “A” as a suffix and its two prefix paths are {C: 1} and {E C B: 1}, which form its conditional pattern base. Its conditional FP-tree contains no node because none of the items in the conditional pattern base can reach the minimum support count of 2. Next for “E”, the conditional patterns base is {B C: 2} and {B: 1}. Its conditional FP-tree contains only one branch <B: 3, C:2>. This branch generates frequent patterns: {B C E: 2}, {B E: 3} and {C E: 2}. For node “C”, the conditional patterns base is {B: 2}, which has the minimum support count. So, its conditional FP-tree is <B:2> and frequent patterns from here is {B C:2}. Finally, for node “B”, there is no conditional pattern base and so, no conditional FP-tree and frequent items.

Item	Conditional pattern base	Conditional FP-tree	Frequent patterns
A	{{(C:1), (E C B:1)}}	Null	A
E	{{(B C:2), (B:1)}}	<B:3, C:2>	B C E:2, B E:3, C E:2, E
C	{{(B:2)}}	<B:2>	B C:2, C
B	None	Null	B

Table 6: Frequent patterns from FP-tree mining

So, this is how FP-growth algorithm generates the list of frequent patterns: {{B}, {C}, {E}, {A}, {B,E}, {C,E}, {B,C} and {B,C,E}}.

### 2.3.3. Frequent Episode Rule:

An episode is a set of sequential transactions in a given period. Briefly, given an event database  $D$ , where each transaction is associated with a time stamp, an interval  $[t_1, t_2]$  is the sequence of transactions that start from  $t_1$  and ends at  $t_2$ . The width of interval,  $w$  is defined as  $t_2 - t_1$ . Given an itemset  $A$  in  $D$ , an interval is a minimum occurrence of  $A$  if it contains  $A$  and none of its proper sub-intervals contain  $A$ . Define  $support(X)$  as the ratio between the number of minimum occurrences that contain  $X$  and the total number of event numbers in database  $D$ . A frequent episode rule is the expression:  $X, Y \rightarrow Z$ ,

[*confidence, support, window*]. Here  $X$ ,  $Y$  and  $Z$  are item sets, and they together form an episode.  $s$  is the support of the rule  $support(X \cup Y \cup Z)$  and  $c$  is the confidence of the rule.  $c = support(X \cup Y \cup Z) / support(X \cup Y)$ . The width of each of the occurrences must be less than window ( $w$ ). When minimum confidence, minimum support and time window are known, this algorithm for episode rules can compute all frequent episode rules [MT96] [LSM99a] [LXY03].

In NIDS it is sometimes very important because some events occur with other events and this is a normal behavior if it occurs within a certain time period. After finding all frequent episodes we can say them as innocent. Since normal behavior occurs more frequently than an abnormal behavior, a frequent behavior will never be an intrusion. As an example, let us consider the SYN flood attack. When launching this attack, an attacker uses many spoofed source addresses to open many connections which never become completely established (i.e. only the first SYN packet is sent, and the connection remains in  $S_0$  state) to some port on a victim host (e.g., http) [LSC+01a]. Figure 14 [LSM99a] [LSM99b] shows such frequent episodes rule and also explains it.

Frequent Episode	Meaning
(service = http, flag = $S_0$ , dst_host = victim), (service = http, flag = $S_0$ , dst_host = victim) $\rightarrow$ (service = http, flag = $S_0$ , dst_host = victim) [0.93, 0.03, 2]	93% of the time, after two http connections with $S_0$ flag are made to host victim, within 2 seconds from the first of these two, the third similar connection is made, and this pattern occurs in 3% of the data

Figure 14: Frequent Episodes Rule (SYN flood attack)

## 2.4. Data Mining based NIDS Projects using Association Rule Mining

Data mining techniques especially Association rule mining was able to get a lot of attention from many researchers from early 2000 in the field of Network Intrusion detection. Some famous projects that implement Association rule mining include ADAM

[BCJ+01], MADAMID [LS00], MINDS [ELK+04], LERAD [MC03] etc. In next few sub-sections we will discuss these algorithms in details.

### 2.4.1. ADAM

ADAM [BCJ+01] was one of the most important research in this field at the time of 2001 and 2002. A lot of research have been done on improvement of this algorithm later on.

ADAM uses a combination of association rules and classification to detect any attack in a TCP Dump audit trails. 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 is described below.

#### 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 consists of attack free connections. The schema of the database is shown in Table 7.

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

**Table 7: 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 7.

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 8: 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  $X \rightarrow Y$ . Suppose, in some specified period of time a rule ( $src\_IP = 137.207.34.1, src\_port = 80 \rightarrow service = http$ ) 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 ( $dest\_IP = 137.207.34.1, dest\_port = 80 \rightarrow flag = SYN$ ) (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.

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. They do not have any mechanism to avoid those redundant and irrelevant rules. *For example, suppose a rule is  $(A,B) \rightarrow C$ , which means that if A and B occurs then C will occur. It already confirms that if B occurs then C will occur. But this algorithm will compute  $B \rightarrow C$  also as a different rule, which means that this algorithm generates unnecessary extra rules.* But later, a lot of studies have been done on this approach and many researchers introduced various measures (like interestingness,  $I$ ) [MC03] [LSF+00] into their consideration and improved this model. Such a model is described in detail in section 2.4.3. Another weakness of ADAM is that it totally depends on attack free normal training data, which are difficult to get.

#### **2.4.2. MADAM ID**

MADAMID is one of the well known IDSs in this field. In this paper [LS00] their aim was to develop a more systematic and automated approach for building IDS. They have developed a set of tools that can be applied to a variety of audit data [section 2.1.2] sources to generate intrusion detection models. The central theme of MADAMID approach is to apply data mining programs to the extensively gathered audit data to compute models that accurately capture the actual behavior or patterns of intrusions and normal activities. The main components of MADAMID framework include learning classifiers and meta-classifiers [Section 2.1.3], association rules [Section 2.2.1] for data analysis and frequent episodes [section 2.3.3] for sequence analysis. The process of applying MADAMID is as follows:

In the first step, raw audit data are gathered in binary format. Then, they are processed into ASCII network packet information. For example, initially they were some bytes in 0 and 1. Then, we convert those values to ASCII format, so that we can easily understand them. Suppose first 16 bit number indicates the source port number, so we convert the first 16 bit binary number into hex or decimal so that we can understand the source port number. After decoding all packet header information we then summarize them into

connection records containing some basic features like service, duration etc. The sample connection records after converting into ASCII will look like Table 8.

Timestamp	Duration	Service	src_host	dst_host	src_bytes	dst_bytes	Flag	...
1.1	0	http	spoofed_1	victim	0	0	S0	...
1.1	0	http	spoofed_2	victim	0	0	S0	...
1.1	0	http	spoofed_3	victim	0	0	S0	...
1.1	0	http	spoofed_4	victim	0	0	S0	...
1.1	0	http	spoofed_5	victim	0	0	S0	...
1.1	0	http	spoofed_6	victim	0	0	S0	...
1.1	0	http	spoofed_7	victim	0	0	S0	...
...	...	...	...	...	...	...	...	...
10.1	2	ftp	A	B	200	300	SF	...
12.3	1	smtp	B	D	250	300	SF	...
13.4	60	telnet	A	D	200	12100	SF	...
13.7	1	smtp	B	C	200	300	SF	...
15.2	1	http	D	A	200	0	REJ	...
...	...	...	...	...	...	...	...	...

Table 9: Sample Network Connection Records

Various data mining programs like association rule, frequent episode rule are then applied to those connection records and as an output they got some derived features and then these derived features are used as rules in models. For example, suppose in connection records we have got that from the same source IP many packets are trying to access to many destination IPs but with same port. In packets/ event records step all these information were discrete and they are brought together on the basis of a certain time window (which was 5 minutes in their experiments) in the connection/ session records step. For example, after applying frequent episode rule [section 2.3.3] into data of Table 8, we get the output like Table 9.

Frequent Episode	Meaning
(service = http, flag = S <sub>0</sub> , dst_host = victim), (service = http, flag = S <sub>0</sub> , dst_host = victim) → (service = http, flag = S <sub>0</sub> , dst_host = victim) [0.93, 0.03, 2]	93%of the time, after two http connections with S <sub>0</sub> flag are made to host victim, within 2 seconds from the first of these two, the third similar connection is made, and this pattern occurs in 3% of the data

Table 10: Example output of intrusion pattern from Table 8

Then, after applying data mining rules (association/ frequent episodes) in these records we come to know a feature that if above condition happens then, this might be an anomaly or attack and we get a rule describing this situation from this step. Then, at last this rule is applied into the model. Since all these data mining methods (association rule, frequent episode rule) are described with example in section 2.2.1 and 2.3.3 accordingly and feature construction is described in 2.1.3., they are not discussed here again.

Currently, MADAM ID produces misuse detection models for network and host systems as well as anomaly detection models for users. The main strength in their approach is that they have focused on efficiency and automated the process of feature constructions. Their limitation is that their system is currently off-line and they are studying how to convert it into real time IDS because effective intrusion detection system should be real time system to minimize the security compromise. Another limitation in this model is that it computes only the frequent patterns of connection records. But many intrusions like those that embed all activities within a single connection do not have frequent patterns in connection data. These types of intrusions might go undetected in their model.

### **2.4.3. LERAD**

This research [MC03] presented an efficient algorithm called LERAD (Learning Rules for Anomaly Detection). They presented it as an alternative to ADAM [BCJ+01] algorithm. The main difference between ADAM algorithm and LERAD algorithm is that ADAM produces all possible association rules and relations and as a result the rate of false alarm is also very high. On the other hand LERAD produces fewer selected rules, which are free of redundancy and the false alarm rate is also lower than in ADAM algorithm. For example, suppose a rule is  $(A,B) \rightarrow C$ , which means that if A and B occurs then C will occur. It already confirms that if B occurs then C will occur. But ADAM will compute  $B \rightarrow C$  as a different rule while LERAD will remove this rule as redundant.

In LERAD, at first, it selects a consequent. Then it starts adding antecedents to create new rules. Finally, it removes rules where rule's antecedent is part of another rule's antecedent. For example, we select  $C$  as consequent. Then we create a rule,  $A \rightarrow C$ . Next, we add another antecedent  $B$  to the rule and it becomes  $(A,B) \rightarrow C$ . In the second phase of LERAD, we see that  $(A,B) \rightarrow C$  has already marked the antecedent  $(A)$  of the rule  $A \rightarrow C$ , so we remove it.

The algorithm and how it produces fewer rules are described with example below.

Suppose  $S$  is the sample of training data sets in Table 10.

Port	Word1	Word2	Word3
80	GET	/	HTTP/1.0
80	GET	/index.html	HTTP/1.0
25	HELO	pascal	

Table 11: Sample Training Dataset

Here Port, Word1, Word2 and Word3 are four attributes.

Step 1: In this step it generates all possible rules and relations from the sample. The algorithm is as follows:

```

Repeat  $L$  times
  Randomly pick two instances  $S_1$  and  $S_2$  from  $S$ 
  Set  $A = \{a: S_1[a] = S_2[a]\}$  (matching attributes)
  For  $m = 1$  to  $M$  and  $A$  not empty do
    Randomly remove  $a$  from  $A$ 
    If  $m = 1$  then create rule  $r_i = "a = S_1[a]"$ 
    Else add  $S_1[a] = a$  to  $r_i$ 's antecedent
  Add  $r_i$  to rule set  $R$ 

```

Figure 15: LERAD Algorithm ([MC03] page: 602)

In the first line of the algorithm in figure 12, we randomly pick two instances  $S_1$  and  $S_2$  from sample  $S$ . So,  $S_1 = \{80, GET, /, HTTP/1.0\}$  and

$$S_2 = \{80, \text{GET}, /index.html, \text{HTTP}/1.0\}$$

Then, we match the attributes of  $S_1$  and  $S_2$  in the second line. The matched attributes are (Port, Word1, Word3).

Then, in the third line we start a loop. The loop goes 1 to M. Suppose in this case let  $M=4$ . Now we enter into the loop in the next line. Here, we randomly choose Word1 as  $a$  from the list of attributes  $A$  and remove it from  $A$ . So now  $a=\text{Word1}$  and  $A=\{\text{Port}, \text{Word3}\}$ . For the first time  $m=1$  and we go inside the *if* statement and create a rule  $r_1$ : **Word1=GET**. We add this rule to the ruleset.

The second time  $m=2$  and attribute set “A” is not empty. So again we go inside the loop. This time we randomly remove another attribute “Port” as “a”. So now,  $a=\text{Port}$ ,  $A=\{\text{Word3}\}$ . This time we go to the *else* part as  $m$  is not equal to 1.  $S_1[\text{Port}] = 80$  and we add this as antecedent of rule2  $r_2$ . So,

**$r_2$ : if Port = 80 then Word1 = GET.** We add this to the ruleset.

Third time still  $m < 4$  and  $A$  is not empty. We randomly choose the only one attribute left Word3 and remove it from  $A$ . Now  $a=\text{Word3}$  and  $A=\{ \}$ . Then, we go to the *else* part.  $S_1[\text{Word3}] = \text{HTTP}/1.0$ . We add this as antecedent of  $r_3$ .  **$r_3$ : if Port = 80 and Word3 = HTTP/1.0 then Word1 = GET.** We add this to the ruleset.

Fourth time  $m=4$  and also  $A$  is empty. So we break the loop. This is how the algorithm generates rules and the whole process continues until generation of all rules. So finally in this step our ruleset is:

$$R = \{$$

$$r_1: \text{Word1} = \text{GET},$$

$$r_2: \text{if Port} = 80 \text{ then Word1} = \text{GET}$$

$$r_3: \text{if Port} = 80 \text{ and Word3} = \text{HTTP}/1.0 \text{ then Word1} = \text{GET}$$

$$\}$$

Step 2: In this step we order those rules in decreasing order and remove any redundant rules. To sort these rules, we use a score of  $n/r$ , where  $n$  is the number of training instances satisfying the antecedent and  $r$  is the number of allowed value. The algorithm is as followed in figure 16:

```
Update the consequents in  $R$  over  $S$ 
Sort  $R$  by decreasing  $n/r$ 
For each rule  $R_i$  in  $R$  in decreasing order of  $n/r$ 
    Mark the values predicted by  $R_i$ 
    If no new values can be marked, remove  $R_i$ 
```

**Figure 16: LERAD algorithm (Part 2)**

For example, in our case, after training over  $S$  and sorting by  $n/r$  these become:

- $r_2$ : if Port = 80 then Word1 = GET ( $n/r = 2/1$ )
- $r_3$ : if Port = 80 and Word3 = HTTP/1.0 then Word1 = GET ( $n/r = 2/1$ )
- $r_1$ : Word1 = GET or HELO ( $n/r = 3/2$ )

As an explanation let's see how  $n$  and  $r$  are selected of  $r_3$ . Number of instances where word1 = GET or HELO is 3 and the allowed values are both of them which is 2. Let's see another example. For  $r_2$ , the number of instances where the rule is matched is 2 (1<sup>st</sup> and 2<sup>nd</sup> rows of table) and the allowed value here is only GET, so  $r = 1$ . Here the arbitrary value of  $r_2$  and  $r_3$  as same and anyone could be on the 1<sup>st</sup> place.

Removing redundant rule:  $r_2$  marks the two GET values in  $S$ . Rule  $r_3$  would mark the same two values and no new values, so we remove it. Rule  $r_1$  marks the HELO in the third instance in addition to the previously marked values, so we keep this rule.

#### **2.4.4. MINDS**

MINDS is one of the most popular NIDS in current days. This system [ELK+04] was developed at department of computer science in University of Minnesota. University of

Minnesota is using this system in their network from 2002 and they are capable of detecting many new attacks as they are launched (examples include “slammer worm”, “NetBus worm” etc).

There are two kinds of anomaly detection techniques, which are supervised and unsupervised anomaly detection. In supervised anomaly detection, given a set of normal data to train on and given a new set of test data and the goal is to determine if the test data is normal or anomalous. In unsupervised anomaly detection system the model attempts to detect anomalous behavior without using any knowledge about the training data. Unsupervised anomaly detection systems are based on statistical approaches, clustering, outlier detection schemes etc. This MINDS [ELK+04] is a kind of unsupervised anomaly detection system.

MINDS uses a suite of data mining techniques to automatically detect attacks against computer networks and systems. The long term objective of MINDS is to address all aspects of intrusion detection. In this paper they have presented details of two specific contributions: (1) an unsupervised anomaly detection technique that assigns a score to each network connection that reflects how anomalous the connection is, and (2) an association pattern analysis based module that summarizes those network connections that are ranked highly anomalous by the anomaly detection module.

The workflow of the MINDS is described below in the figure 17. We will describe the system with the workflow step by step.



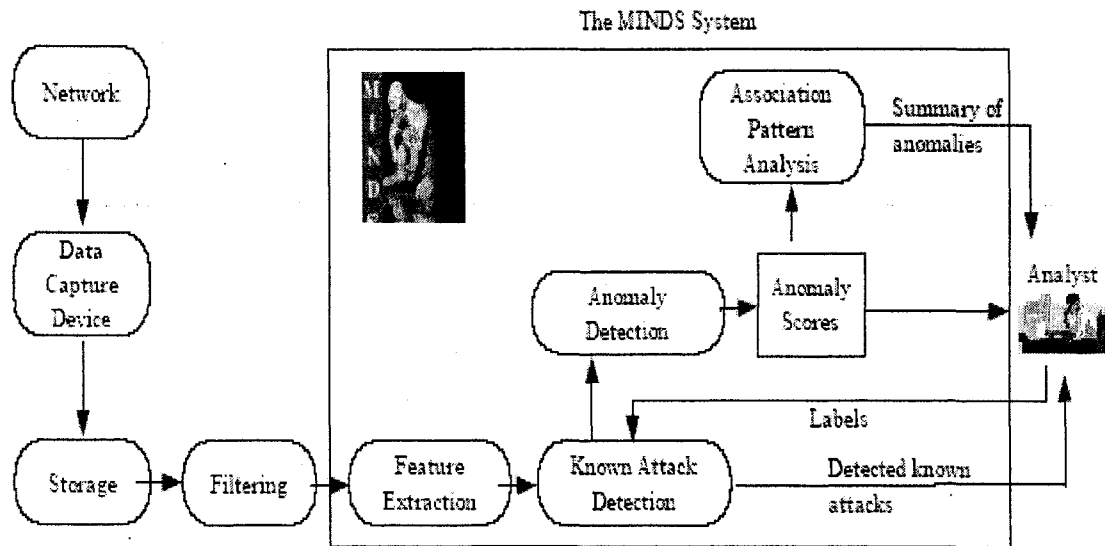


Figure 17: MINDS System

**Input:**

Input to the MINDS is Netflow version 5 data collected using flow tool [FT07] 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 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 connection with similar characteristics within 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 11 and table 12.

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 12: 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 13: 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 sends 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) [BKN+00]. 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 will be 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.

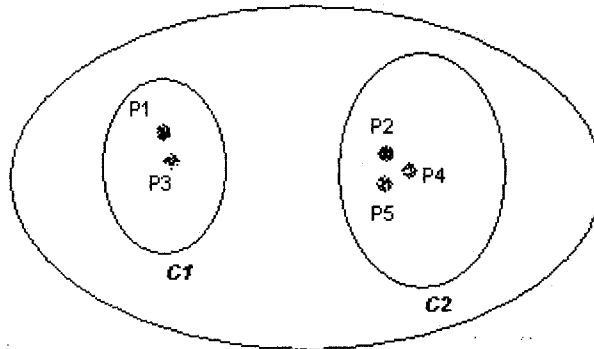


Figure 18: Local Outlier Factor (LOF) approach

For example, in the figure 18 we can see that cluster  $C_2$  is denser than  $C_1$ . Due to the low density in cluster  $C_1$ , for most examples  $q$  inside  $C_1$ , the distance between any dataset and its neighbor is greater than that of  $C_1$ . For example, the distance between  $p_1$  and  $p_3$  is higher than the distance between  $p_2$  and  $p_4$ . So, therefore  $p_2$  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 [Section 2.1.1.] 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 new attack signatures. This is how the MINDS works 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, which is the data section of TCP packets. As a result, U2R or R2U attacks may go undetected in their system.

## **2.5. Wireless Intrusion Detection System (WIDS)**

Wireless Intrusion Detection Systems (WIDS) will monitor a WLAN using a mixture of hardware and software called sensors [Hu04]. WLAN IDSs can be host based, network based and hybrid. Both host based and network based IDSs are equal from a central control perspective. WLAN network based IDSs are usually deployed at centralized administration points (e.g., WAP) to monitor the WLAN network traffic [LLM+07].

Typically, all wired network attacks from network layer and above can be used on WLANs and therefore most intrusion detection techniques for wired networks can be applied to WLAN IDSs [LLM+07].

Rogue WAPs are specific threats to WLAN. In general, rogue (unauthorized) WAP detection follows a two-step process: 1) identify whether it is present, 2) decide whether it is rogue. Radio frequency (RF) scanning is the most common technique used by WLAN IDSs to detect rogue WAPs. Once a WAP is discovered, a pre-configured authorized list of WAPs is used to identify whether it is rogue. Any newly detected WAP that falls outside the authorized list would be considered rogue [LLM+07].

Currently, there are many open source wireless scanners. These scanners use Radio Frequency (RF) to detect any new WAP in the vicinity and can monitor their traffic. With these sensors we can get a picture of the wireless networks around. Examples of these scanners are Kismet [Hu04], NetStumbler [Sh04], airSnort [Air07c]. Then, this information can be used to create a WIDS. To understand these network traffic and extracting alerts from it we need professional security analyst, who can interpret the alerts and make sense of the output. In misuse WIDS, a list of attack signature is kept to detect any attack. We have seen that at a pre-configured authorized list of WAPs is also maintained to detect rouge WAPs. In a similar way, we also need to maintain a list of authorized MAC addresses to detect any rouge MAC or MAC spoofing. Commercial WIDSs offer all these features in an integrated single software. The major benefits of wireless IDS technology that would enhance the defensive posture can be categorized in two different groups [Me06]:

- Real-time Network Monitoring and Radio Frequency (RF) Management: Wireless IDS can monitor network activities (e.g., packets coming into the network, packets going outside from the network, clients connected etc.) in real-time and also it can control/ manage to which channel it will broadcast or operate its network packets or activities.
- Intrusion Detection and Response: Wireless IDS has the capability to detect intrusions in wireless environment and can block these suspicious connections or pass them to analyst for further review.

These functions can be partly done by wired network IDS, but a wired IDS does not have Radio Frequency (RF) management function although it can do the real time wired network monitoring.

Examples of such enterprise software include AirMagnet [Air08a], AirDefense [Air08b], Red-M [Red08], AirWave [Air08c], BlueSocket [Blu08] etc.

## **2.6. Data Mining in WIDS**

Wireless Intrusion Detection is a newer area of research and still it is getting a lot of attention among researchers and industry communities. So far, works on wireless intrusion detection have focused on improving the architecture or protocol of WLANs and on detecting specific types of attacks [BS03] [HPJ03] [WZS02]. But more recently there are some research on how data mining concept can be applied to detect anomalous traffic in wireless networks including WLAN and ad-hoc networks [ZKN05] [LLM+07].

### **2.6.1. A Clustering Approach to Wireless Network Intrusion Detection**

In this research [ZKN05], authors have analyzed network traffic data streams collected and recorded from a WLAN system and in detecting all types of attack behaviors through data mining technique specially clustering technique. The log they have used here is specifically for wireless traffic and they have extracted these data from several access points (APs). The metrics they have studied for characterizing wireless network attacks include Broadcast SSID, Sequence number of AP etc., which cannot be found in normal wired TCP traffic. In their approach they have clustered wireless traffic data and used heuristics to label each instance as intrusive or normal. The heuristic is simple, where clusters are ordered according to the distance to the largest cluster and a percentage cutoff is used to determine the separation point between attacks and normal clusters. The assumption they have used is that since normal instances are generally very dominant in the collected data, the largest cluster is usually composed of normal instances and anomalous or attack instances would belong to clusters that are far away from the largest

cluster. In their system, they have used online-k-means algorithm, which is claimed to outperform the standard k-means algorithm.

The wireless logs they have used for this research are available at <http://nms.lcs.mit.edu/~mbdazin/wireless/>. It corresponds to wireless trace collection from a real network with more than 170 access points spread over three physical locations (buildings) over a period of several weeks. The wireless network used for data collection was operating in the infrastructure mode with clients connected via the access points.

After collecting the raw data from wireless logs, they have preprocessed the data to make it suitable for clustering algorithm. In this study, they were constrained by the metrics that were available in the recorded wireless logs rather than having all the metrics that are theoretically required to model common wireless attacks. The metrics used in the study are described in table 14:

Metric	Description
AID	The highest occurring (most occurrences) value will be used for the set of records that are grouped if a non-unique value is found. Over all there will be 4 categories.
Parent	A categorical value representing 173 distinct access points. This value will be retained along with a distinct MAC address pair, it shows connection related activity.
Day	A categorical value in {Weekday, Weekend}.
Time_slot	A categorical value in {Morning, Day, Evening}, to represent the peak and off-peak time periods.
ShortRet	A numeric value averaged across groupings.
LongRet	A numeric value averaged across groupings.
Quality	A numeric measure in [0, 100], averaged to give the idea of mean quality.
Strength	Signal strength, a numeric value in [0, 100].
SrcPkts	The number of packets a station sourced, which could be averaged to show the transmission activity level of the station. It can later be discretized into low, medium and high.
SrcErrPkts	Number of errors in source packets.
DstErrPkts	Number of errors in destination packets.
DstMaxRetryErr	Number of observed max-retry error packets for which this station was the destination.

**Table 14: Metrics used to cluster the wireless logs**

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. Suppose table 15 represents the input data.



Site	Day	Moment	AP	SysUpTime	SnmpInPkts	SnmpOutPkts	IpIn	IpOut	IpFwd	TcpIn	TcpOut	UdpIn	UdpOut
LBdg	2.7.2 0	00:05:0 7	AP 1	3:1:57:12. 0	523976	522317	984212	528030	0	1911	1911	523187	524183
MBdg	2.7.2 0	00:10:0 7	AP 2	3:2:02:12. 0	524444	522783	984744	528498	0	1911	1911	523653	524649

Table 15: 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 author's 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 14, to make use of only five attributes from the sample input as shown in Table 16.

Connection Record	Attribute 1	Attribute 2	Attribute 3	Attribute 4
Con <sub>1</sub>	5	3	2	1
Con <sub>2</sub>	6	6	3	1

Table 16: Sample connection records for kmo algorithm

Each record Con<sub>1</sub>, Con<sub>2</sub> 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

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

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

centroid of cluster  $y_n$ .  $\left\| \cdot \right\|$  represents  $L2$  norm.

Let  $L_i$  and  $M_j = \text{Con}_1, \text{Con}_2$  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  $C_1$  and  $C_2$  at 0<sup>th</sup> iteration

Then the distances between the columns in Table 1 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 17.

$C_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{array}{l} \text{group 1} \\ \text{group 2} \end{array}$
--

**Table 17: 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:

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

$\Sigma$  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.

<u>L<sub>i</sub></u>	<u>M<sub>i</sub></u>	<u>C<sub>1</sub></u>	<u>C<sub>2</sub></u>
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 iteration 1

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

$$D_1 (M_j, C_1) = \sqrt{\{(6-2.95)^2 + (6-2.95)^2 + (3-2.95)^2 + (1-2.95)^2\}} = \sqrt{22.41} = 4.73$$

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

$D_1$  represents distance at iteration 1.

$$D1 = [4.73 \quad 5.83]$$

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

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

**Table 18: 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.

### **2.6.2. A Hybrid IDS for Wireless Intrusion Detection using Association Rule Mining**

In [LLM+07] a novel hybrid anomaly detection approach is proposed which incorporates the association rule mining technique and cross feature mining to build normal behavior profiles of network activities for an individual node. The association-rule mining technique is applied on data collected on cross-layer features, while the cross-feature mining is applied on data collected on statistical features.

The proposed system is built on four major components: data collection module, profile module, detection module and decision module.

#### **Data Collection Module:**

This module collects network data according to the proposed two feature sets within radio transmission range. The two proposed feature sets are cross-layer feature set and statistical feature set. Examples of cross layer feature set are sourceIP, destinationIP, MAC frame type (RTS/ CTS/ DATA/ ACK), flow direction (SEND/ RECV/ DROP) and type of packet (data/ control). Examples of statistical feature set are time, inbound traffic data, outbound traffic data, transmit traffic rate or receive traffic rate. For example, suppose in a specified duration the data collection module collects the following data specified in Table 19 and Table 20.

Time	Inbound/ Outbound	Transmit/ Receive Traffic rate where r is specified threshold
T1	Outbound	Transmit < r
T2	Outbound	Transmit < r
T3	Inbound	Receive < r
T4	Outbound	Transmit > r

Table 19: Example of data collected over statistical feature set

Source IP	Dest. IP	MAC frame type	Flow Direction	Packet Type
IP1	IP2	DATA	SEND	DATA
IP1	IP2	DATA	SEND	DATA
IP2	IP1	ACK	RCV	CTR
IP3	IP1	CTS	SEND	CTR

Table 20: Example of sample data collected over cross-layer feature set

**Profile Module:**

There are two subsystems in this module, one is a pre-processor and the other is a profiler. The pre-processor transforms training data into market basket format. Then, the profiler uses Apriori algorithm to find association patterns (rules) from the market basket data. Each test data event (i.e. a transaction record in the converted market basket data) can then be classified according to the normal profile in the succeeding anomaly detection module. For illustration and example, the preprocessor transforms Table 19 into Table 21, which is in market basket format and Table 20 is transformed into Table 22. In Table 21, 'O' represents outbound traffic, 'I' represents inbound traffic, 'I<sub>1</sub>' represents the instance where transmit/ receive rate < r and 'I<sub>2</sub>' represents the instance where transmit/ receive rate > r.

Connection ID	Features
C001	T <sub>1</sub> , O, I <sub>1</sub>
C002	T <sub>2</sub> , O, I <sub>1</sub>
C003	T <sub>3</sub> , I, I <sub>1</sub>
C004	T <sub>4</sub> , O, I <sub>2</sub>

Table 21: Market basket format of Table 19

Now suppose source IP  $IP_1$  is represented by A,  $IP_2$  by B,  $IP_3$  by C; destination IP  $IP_1$  by D,  $IP_2$  by E; in MAC frame type DATA frame by F, ACK frame by G, CTS frame by H; in flow direction SEND by S, RCV by R and in packet type data packet by M and CTR packet is represented by N, then the market basket format of Table 20 can be represented by Table 22.

Connection no.	Features
C001	A, E, F, S, M
C002	A, E, F, S, M
C003	B, D, G, R, M
C004	C, D, H, S, N

**Table 22: Market basket format of Table 20**

After pre-processor module converts the data into market basket format, then the profiler applies Apriori algorithm into the market basket data. For example, suppose Apriori algorithm is applied into Table 18 with support rate 50% and confidence 50% and we get a rule like  $O \rightarrow I_1$  and from Table 21 we can get a rule like  $F \rightarrow M$ , which is actually MAC frame: Data  $\rightarrow$  Packet type: DATA. Now these rules are forwarded to the next module, which is detection module.

#### **Detection Module:**

Anomaly detection is to detect deviance from the norm. In other words, it discovers previously unknown behavior patterns. For association-rule mining, rules extracted from test data are compared with the rules in the expected normal profile. Any new rule or rule with deviations beyond the corresponding support and confidence threshold intervals is considered as an anomaly rule. After detection, profiles are updated accordingly. For example, in the normal profile, if the rule  $O \rightarrow I_1$  is not present, then it could be detected as an anomalous rule.



### **Decision Module:**

In association-rule analysis, a detecting node can trigger a local alert when it detects anomaly rules with high support and confidence values. The detecting node can then send a global alert to warn its neighbors. When a detecting node detects anomaly rule with low support and confidence values, it can engage a collaborative decision-making process that incorporate intelligence (global alerts) from its neighbors. For example, if the anomalous rule  $O \rightarrow I_1$  exceeds the specified support and confidence threshold then the system would issue an alert and will update the main profile (which means global alert). If the support and confidence rate is low then it would compare it with other rules found from other sensors and then will issue an alert if necessary.

In this research [LLM+07] a novel hybrid anomaly intrusion detection approach was proposed for wireless ad-hoc networks. The author developed a prototype system to show that the proposed approach can detect a variety of attacks as long as they cause MAC layer misbehavior and/ or network layer misbehavior.

Current wireless IDSs are still dependent on training data and without prior training these systems cannot detect intrusions in real time and some wireless IDS based on Association rule mining technique [LLM+07] detect intrusions only for ad-hoc network and not applicable for infrastructure based WLAN. Some other wireless IDS [ZKN05] used clustering technique which is heavily dependent on a lot of calculation and also converting all connection records to an appropriate axis point is also critical. So, in our thesis we have focused on these issues and derived an effective approach, where we no longer need any training data and our Online Apriori based algorithm is easy to understand and implement while providing the efficiency and accuracy.

### **3. PROPOSED SOLUTION FOR WIRELESS INTRUSION DETECTION**

This chapter gives details of the proposed algorithm and the system workflow used in WiFi-Miner, which can detect anomalies in Wireless LAN successfully. It uses Association rule mining technique based on the Apriori [ISA93] algorithm, which is the core of this thesis. To get the wireless traffic we have used wireless hardware sensors from NetworkChemistry [NC07] vendor. The details on why we choose Apriori based system are discussed in section [2.2.5].

Association rule has been used extensively in the field of network intrusion detection from early 2000. Since then Apriori algorithm has got a lot of attention among the researchers. All the work that involve Association rule mining in Network Intrusion detection deal with Apriori algorithm: with the help of Apriori or its improved version [SON95] [HPY+04] [MC03][Yo03], they, at first detect frequent patterns from a safe database, then train the classifiers with these frequent patterns. After that they check any incoming network connections with these safe frequent patterns, if there is a match it is safe otherwise it is an anomaly.

In this thesis, we proposed an algorithm called WiFi Miner, which finds infrequent patterns in parallel of finding frequent patterns without any training phase. Instead of comparing each incoming record with previously found frequent patterns from the training phase, our proposed system assigns an anomaly score to each record based on the presence of frequent and infrequent patterns in that record. Connection records with positive anomaly score have more infrequent patterns than frequent patterns and are flagged as anomalous packets on the fly.

The rest of the chapter is organized as follows: section 3.1 gives an overview of the proposed system, section 3.2 discusses the Input and Preprocessor module, section 3.3 focuses on Anomaly Detection module and section 3.4 provides a simple example application of the Apriori-Infrequent and anomaly Score Calculation.

### **3.1. Overview of the proposed System**

The proposed WiFi Miner system framework comprises of three main modules [REA08]<sup>1</sup>. They are: Input Module, Preprocessor Module, and Anomaly Detection Module as shown in figure 21. The proprietary Network Chemistry wireless hardware sensors [NC07] first need to be properly installed and configured before they can be used to capture wireless network packets. Installing the sensors entails installing both a sensor server and sensor client software systems and logging on to the sensor client software console system to initialize and configure the sensors. Input Module consisting of properly configured hardware sensors, collects network traffic data from hardware wireless sensors attached to the system, which capture data from airwaves as most of the wireless attacks may occur before data are in wired network and Access Points. The Preprocessor Module converts the raw data to readable format with the help of CommView for WiFi [CV07] software, which is used to extract sensed data from the hardware sensor's firebird database and saved in a .csv file (csv stands for Comma Separated Values where attributes values are simple text separated by commas). With CommView, necessary features can be extracted for analyses to detect anomalies and extracted records stored as text file are processed directly by our WiFi Miner system. These records may also be logged into database tables for more offline processing and possible tracking of anomalous records. The focus of our approach is online processing, that is independent of training data. After the data are preprocessed, they are sent to the Anomaly Detection Module, which includes the core algorithm (Apriori-Infrequent) for finding infrequent patterns or anomalies.

---

<sup>1</sup> This paper [REA08] is from this thesis work.

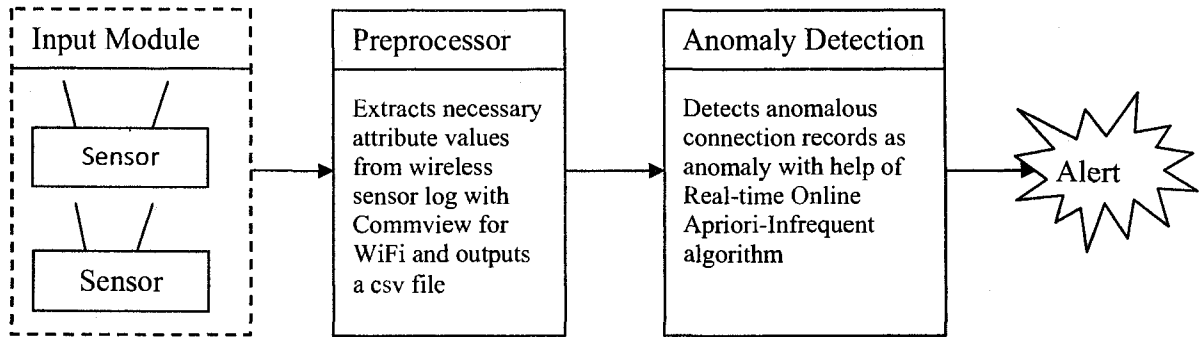


Figure 21: System workflow

The proposed Online Apriori-Infrequent algorithm contributes by

1. Providing a mechanism for computing the anomaly scores of a record, that is based on the relative sizes and numbers of infrequent and frequent itemsets contained in just this record without the need for hard-to-get training data. This is based on the premise that infrequent itemsets are likely anomalous as is the case with many wireless attacks.
2. Providing a smart-join mechanism that improves the Apriori-gen join step and prune steps when computing candidate itemsets, which speeds up infrequent and frequent pattern generations.
3. Providing a mechanism that eliminates the need to generate association rules from frequent patterns in order to detect anomalies.

The WiFi Miner algorithm is presented as Algorithm 1 in figure 22. The proposed scheme finds anomaly/infrequent patterns without training classifiers offline with safe data. Instead of finding frequent patterns at first and then comparing these patterns with incoming data to detect the anomalies during third step, our method finds the infrequent data/anomalies during the first step with an online Apriori-Infrequent algorithm, which tries to find both infrequent patterns and frequent patterns, improves candidate set generation scheme in one step by improving the runtime complexity of Joining and Pruning. The rest of the chapter describes both the Online Apriori-Infrequent algorithm and the Anomaly scoring scheme adopted by the proposed WiFi Miner system.

**Algorithm 1. (WiFi Miner: Wireless IDS)**

**Algorithm WiFi Miner()**

**Input:** Network connection packets (P), sensors (S), access points (AP)

**Output:** Anomalous connections (A)

**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 Apriori-Infrequent Algorithm with  
“Incoming-connection” .csv file records as input  
and output anomalous records (A) as alerts.*

**end**

Figure 22: Algorithm 1 – WiFi Miner Workflow

### 3.2. **Input and Preprocessor Module**

Our Input module contains the proprietary Network Chemistry wireless hardware sensors [NC07] and Preprocessor module contains the Commview for WiFi [CV07] software. The goal of the input module is to capture wireless network packets successfully from a selected Access Point (AP) and log them into the hardware sensor’s Firebird database. Then the preprocessor module converts the raw data to readable format with help of CommView for WiFi software and outputs a csv file. So, for a functional Input and Preprocessor module we need to properly install and configure 1) Wireless Access Point, 2) Wireless hardware sensors (both a sensor server and a sensor client software), 3) CommView for WiFi software as shown in figure 23.

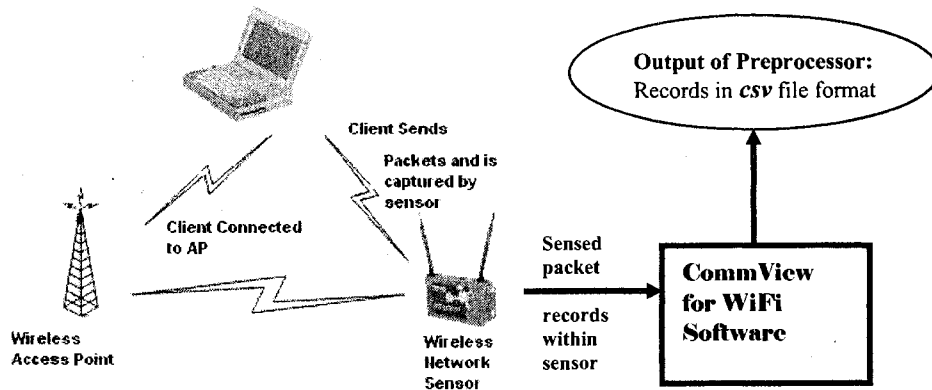


Figure 23: Input Module and Preprocessor Module

Clients connect to the wireless network through Access Point 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.

### 3.2.1. Network Chemistry Sensor Software Installation Process

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 24, and then we enter the password.

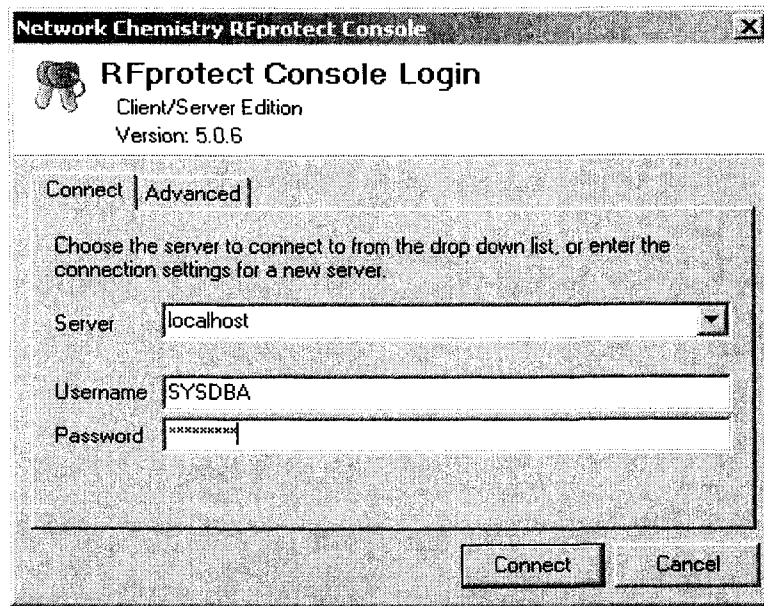


Figure 24: RFProtect Login Window

The screen in figure 25 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 26 is open.

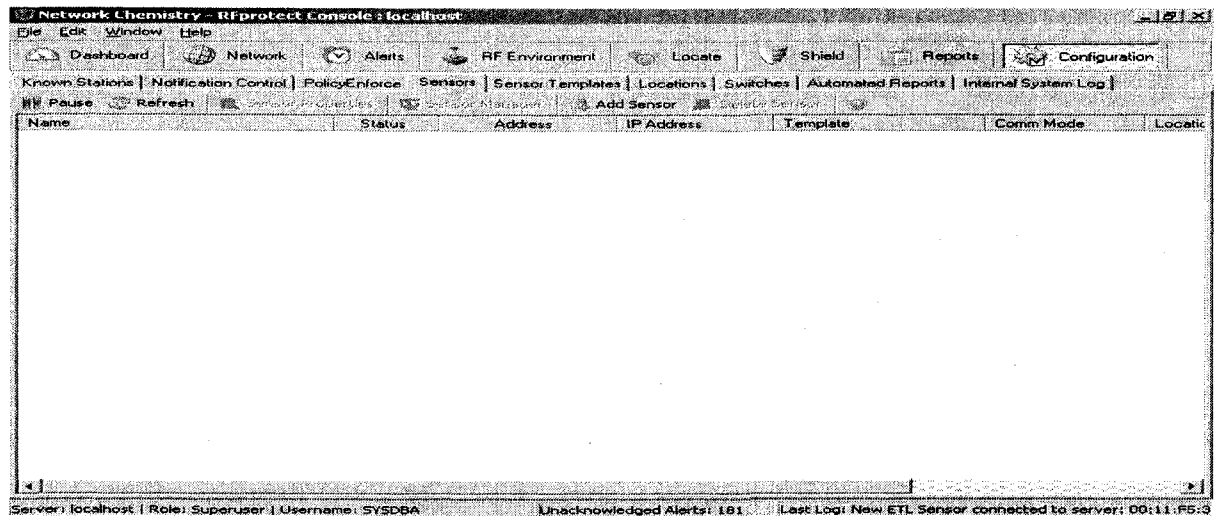


Figure 25: Sensor Configuration Window



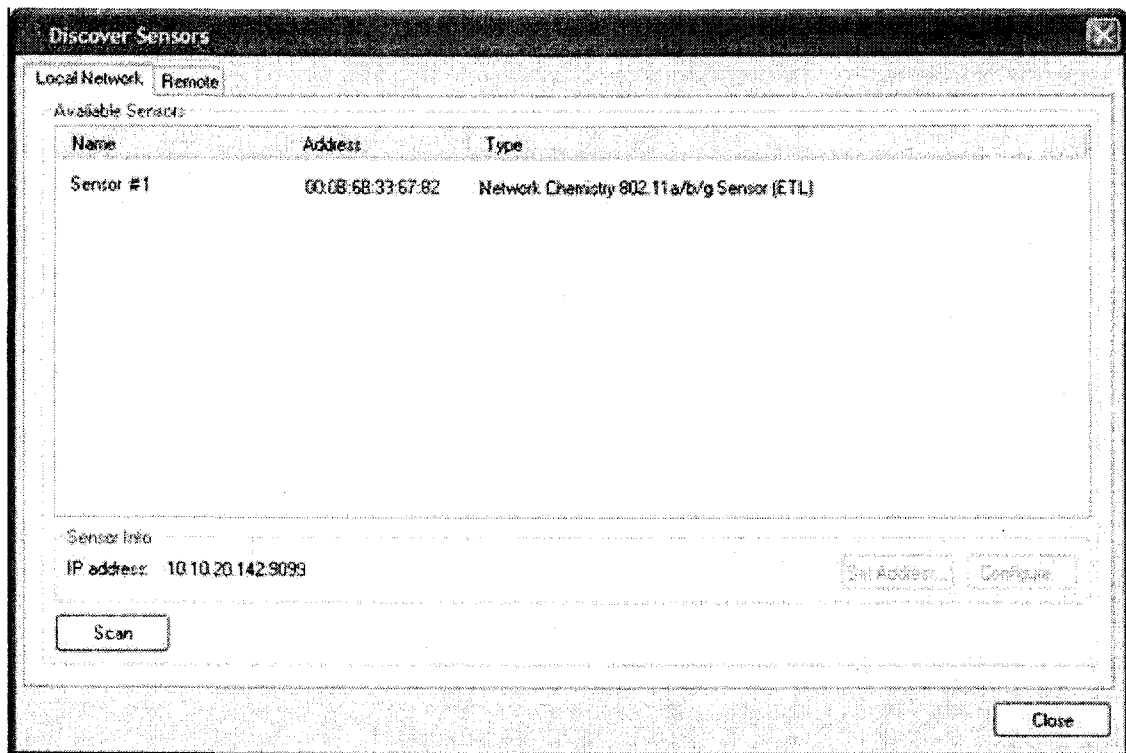


Figure 26: Discover Sensor Window

We double-click our Sensor that we want to configure. The dialog in figure 27 appears.

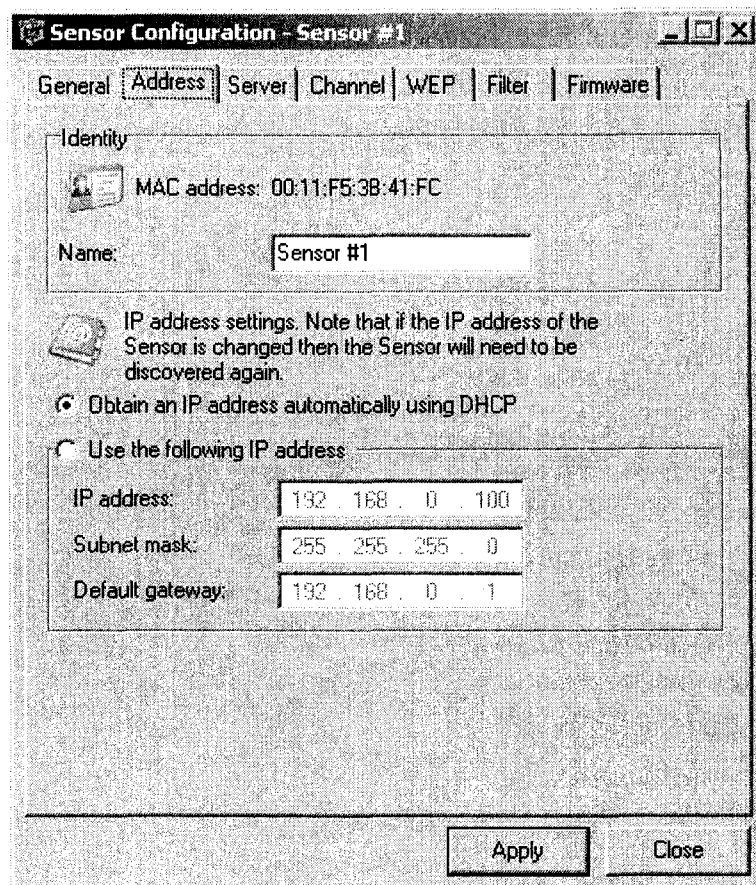


Figure 27: Sensor IP address Configuration

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 28. 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.

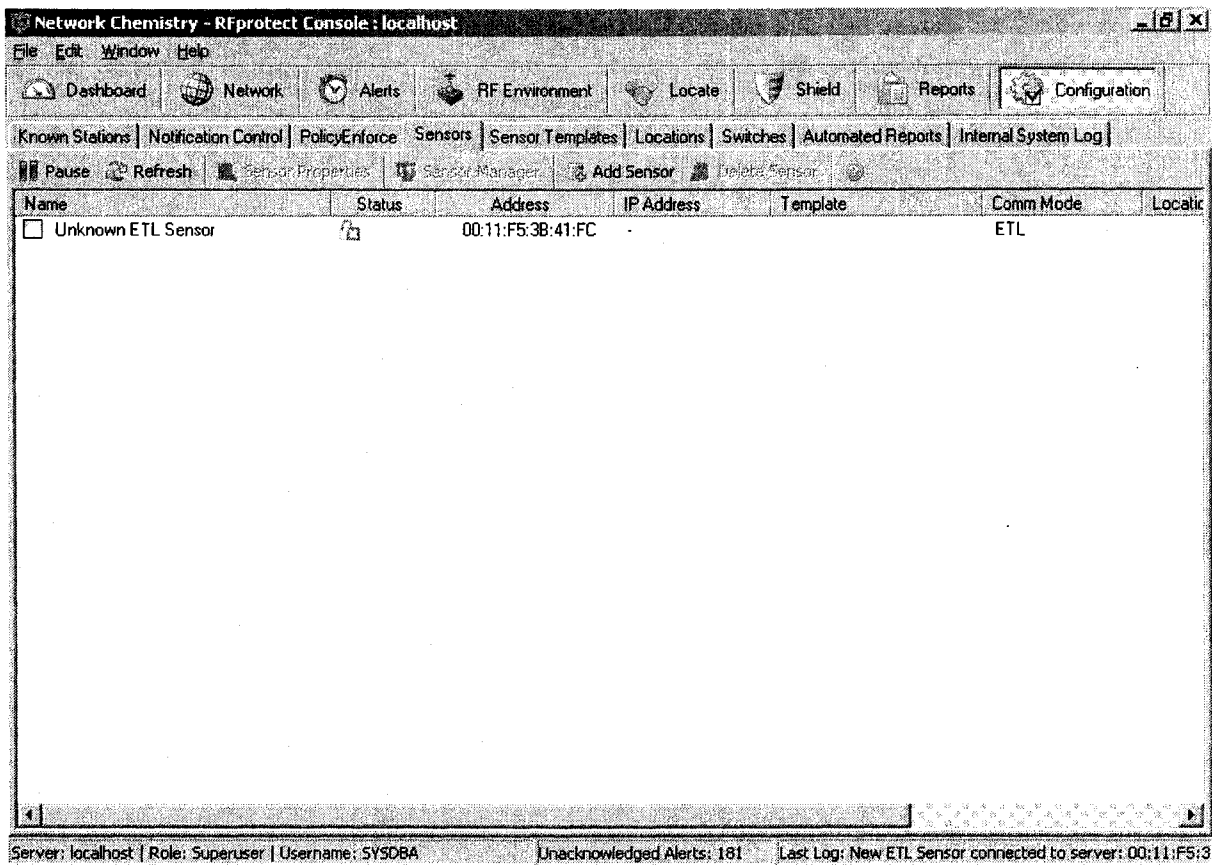


Figure 28: Console window with unknown sensor

Once the sensor is configured to send data to the server, the sensor is displayed in the Console. Figure 29 shows our sensor in a communication state.

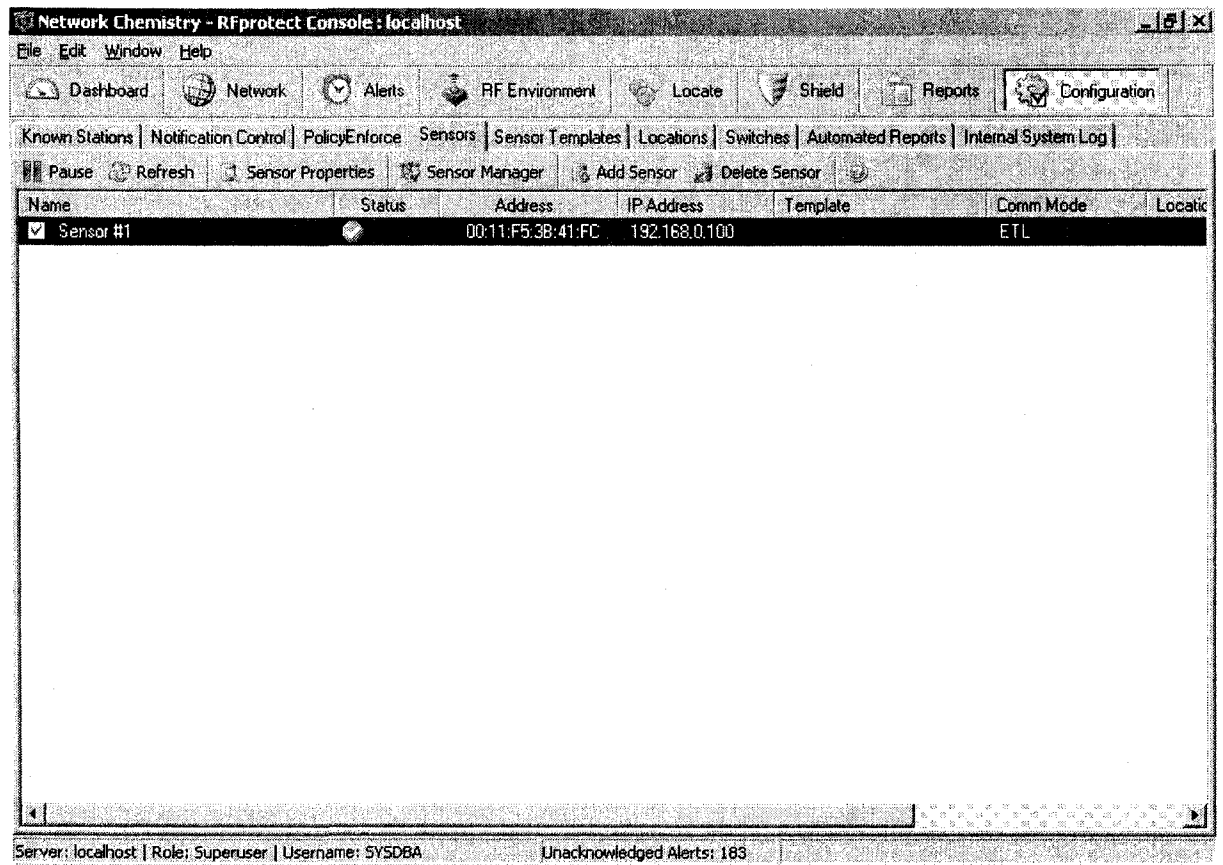


Figure 29: Console window with sensor

### *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 30 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 30.

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 31 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.

# WiFi Miner: An Online Apriori and Sensor Based Wireless Network Intrusion Detection System

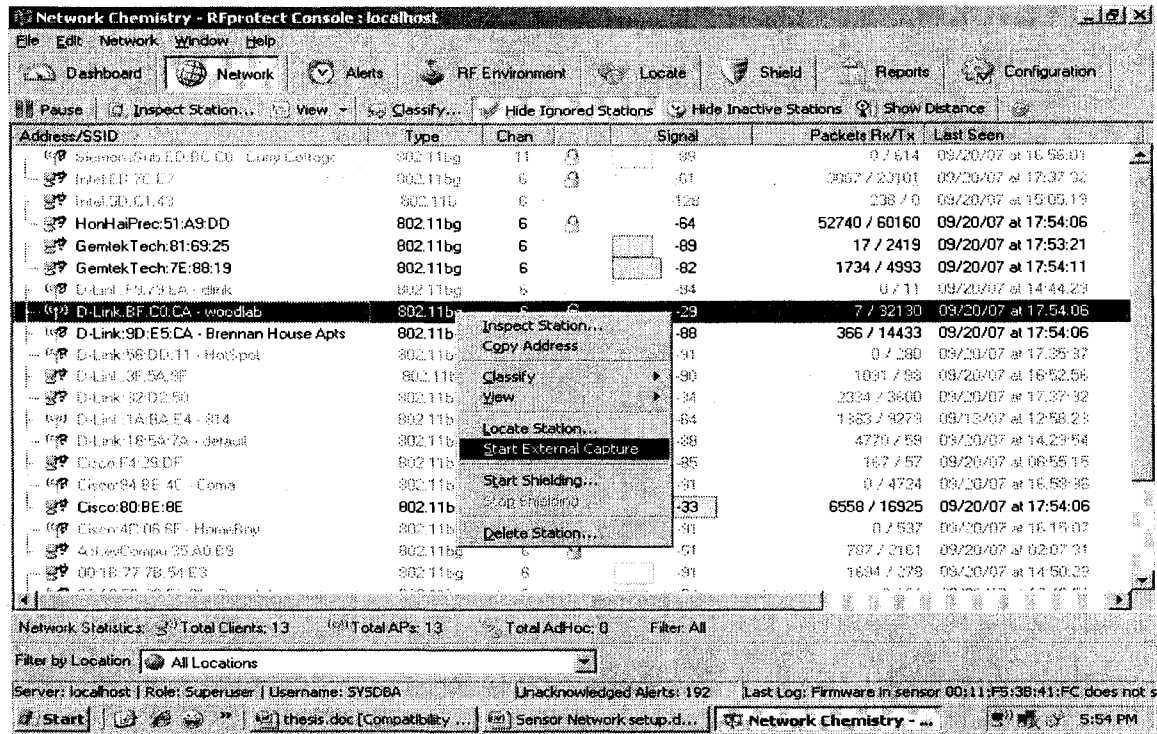


Figure 30: Console with access points and clients

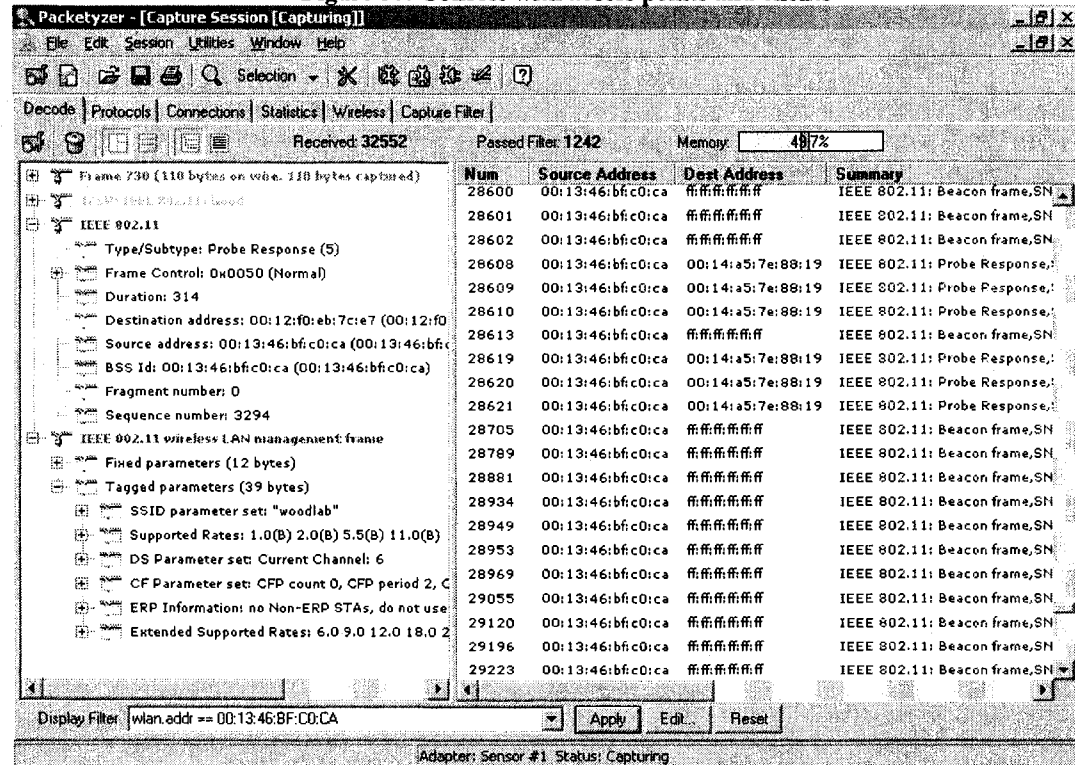


Figure 31: Captured packets using Packetyzer

### 3.2.2. Preprocessing phase using Commview for WiFi

After wireless packet records are successfully sensed and stored by Network Chemistry sensors in its server database, then we can use Commview for WiFi software to export these packet records into a csv file. Records in csv file are already preprocessed, cleaned and organized. A sample of csv file is shown in figure 32.

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	IntelD8:6	Broadcast	10.0.35.22	10.0.1.1	N/A	N/A	23:32.7	74	6	6	1 WEP/WPA: Not encrypted
843	842 IP/ICMPv4	00:18:77:8 33:33:00:0	fe80::25d5ff02::0016	N/A	N/A	N/A	N/A	23:32.7	108	71	71	1 WEP/WPA: Not encrypted
844	843 IP/UDP	00:18:77:8 33:33:00:0	fe80::25d5ff02::0001	52327	5355	1900	1900	23:32.7	106	71	71	1 WEP/WPA: Not encrypted
845	844 IP/ICMPv4	HonHaiPri 33:33:FF:0	fe80::4184ff02::0001	N/A	N/A	N/A	N/A	23:32.7	104	75	75	1 WEP/WPA: Not encrypted
846	845 MNGT/BE	ArubaNet Broadcast	N/A	N/A	N/A	N/A	N/A	23:32.7	98	11	11	1 uwindor(infra.), Ch.#11
847	846 IP/UDP	00:1E:52:7 01:00:5E:7	192.168.1:239.255.25	1900	1900	1900	1900	23:32.7	404	25	25	11 WEP/WPA: Not encrypted
848	847 CTRL/ACK	N/A	00:1E:52:7 N/A	N/A	N/A	N/A	N/A	23:32.7	10	3	3	2 WEP/WPA: Not encrypted
849	848 IP/UDP	00:1E:52:7 01:00:5E:7	192.168.1:239.255.25	1900	1900	1900	1900	23:32.7	404	26	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	N/A	23:32.7	10	5	5	2 WEP/WPA: CRC
851	850 MNGT/BE	Enterasys Broadcast	N/A	N/A	N/A	N/A	N/A	23:32.7	73	3	3	2 CS-WL-2(infra.), Ch.#11
852	851 IP/UDP	00:1E:52:7 01:00:5E:7	192.168.1:239.255.25	1900	1900	1900	1900	23:32.7	404	6	6	2 WEP/WPA: CRC
853	852 IP/ICMPv4	HonHaiPri 33:33:FF:0	fe80::4184ff02::0001	N/A	N/A	N/A	N/A	23:32.7	104	3	3	1 WEP/WPA: CRC
854	853 CTRL/CTS	N/A	HonHaiPri N/A	N/A	N/A	N/A	N/A	23:32.7	10	3	3	2 WEP/WPA: CRC
855	854 CTRL/CTS	N/A	HonHaiPri N/A	N/A	N/A	N/A	N/A	23:32.7	10	6	6	2 WEP/WPA: Not encrypted
856	855 CTRL/CTS	N/A	HonHaiPri N/A	N/A	N/A	N/A	N/A	23:32.7	10	6	6	2 WEP/WPA: Not encrypted
857	856 MNGT/BE	ArubaNet Broadcast	N/A	N/A	N/A	N/A	N/A	23:32.8	96	70	70	1 uwindor(infra.), Ch.#11
858	857 IP/UDP	00:18:77:8 33:33:00:0	fe80::25d5ff02::000c	1900	1900	1900	1900	23:32.8	588	70	70	1 WEP/WPA: Not encrypted
859	858 IP/ICMPv4	GemtekTe 33:33:FF:1	fe80::9416ff02::0001	N/A	N/A	N/A	N/A	23:32.8	104	70	70	1 WEP/WPA: Not encrypted
860	859 IP/ICMPv4	HonHaiPri 33:33:FF:0	fe80::95c5ff02::0001	N/A	N/A	N/A	N/A	23:32.8	104	71	71	1 WEP/WPA: Not encrypted
861	860 IP/UDP	00:18:77:8 33:33:00:0	fe80::25d5ff02::0001	52327	5355	1900	1900	23:32.8	106	70	70	1 WEP/WPA: Not encrypted
862	861 IP/ICMPv4	00:1C:26:3 33:33:FF:5	fe80::411c ff02::0001	N/A	N/A	N/A	N/A	23:32.8	104	71	71	1 WEP/WPA: Not encrypted
863	862 IP/ICMPv4	00:1C:26:3 33:33:FF:5	fe80::411c ff02::0001	N/A	N/A	N/A	N/A	23:32.8	104	68	68	1 WEP/WPA: Not encrypted
864	863 MNGT/BE	ArubaNet Broadcast	N/A	N/A	N/A	N/A	N/A	23:32.8	98	11	11	1 uwindor(infra.), Ch.#11
865	864 DCS(Unk)	01:14:A5:48F:15:66:2	N/A	N/A	N/A	N/A	N/A	23:32.8	80	3	3	11 WEP/WPA: CRC

Figure 32: Sample preprocessed csv file output by Commview for WiFi

Our preprocessor, Commview for WiFi, is a powerful wireless network monitor and analyzer for 802.11 a/b/g/n networks to pre-process captured wireless Sensor logs. With Commview for WiFi, we can easily select which features/ attributes to be exported into the csv file. In preprocessing phase, selecting the right features/ attributes is very important. 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 T interval, will play a vital role in detecting the attacks. After a detailed study of network attacks we have selected the following feature/ attributes that we hope will be able to detect wireless attacks.

<b>Feature/ Attribute</b>	<b>Definition</b>
Frame Type/Subtype	It can be management, control, or data
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 [1 11]

**Table 23: Selected attributes/ features list for preprocessing**

### **3.3. Anomaly Detection Module**

The core part of the WiFi Miner is Anomaly Detection Module. Once wireless connection records are preprocessed then these preprocessed data (csv file) is sent to the Anomaly Detection module. This module includes the core algorithm “Online-Apriori-Infrequent” and mechanism to assign an anomaly score to each record to detect the infrequent patterns or anomalies. Next three sub-sections: 3.3.1 describes the definition and terminology used in Apriori-Infrequent algorithm, 3.3.2 describes the Apriori-Infrequent algorithm and 3.3.3 describes the anomaly score calculation method.

#### **3.3.1. Definition and Terminology**

The following definitions and properties are used in the discussion of the proposed IDS system.

**Definition 1.** A record has a maximal level of  $n$ : if the record,  $R_i$ , has its largest frequent itemset being an  $n$ -itemset or containing  $n$  distinct items. ■

**Definition 2.** A maximal level  $n$  record has a set of frequent and infrequent itemsets: consisting of all its 1-itemsets to  $n$ -itemsets that are frequent and infrequent respectively. ■

**Definition 3.** A Frequent  $k$ -itemset: is a  $k$ -itemset which has support greater than or equal to the given minimum support with respect to the entire database stream of records. ■

**Definition 4.** An Infrequent  $k$ -itemset: is a  $k$ -itemset which has support less than the given minimum support with respect to the entire database stream of records and has all its subsets frequent in levels  $k-1$  and lower. This type of itemset is also called negative border in some work. ■

**Definition 5.** A maximal level  $n$  Record's Frequent Itemsets,  $FR$ : consists of the set of all its 1-itemsets to  $n$ -itemsets, which have supports greater than or equal to the given minimum support with respect to the entire database stream of records. ■

**Definition 6.** A maximal level  $n$  Record's Infrequent Itemsets,  $IFR$ : consists of the set of all its 1-itemsets to  $n$ -itemsets, which have supports less than the given minimum support with respect to the entire database stream of records. All subsets of each level  $k$  Infrequent set are frequent in the levels  $k - 1$  and lower. ■

**Definition 7.** A  $k$ -itemset Anomaly Score: The anomaly score of a level  $k$  itemset is  $-k$  if the itemset is frequent but  $+k$  if the itemset is infrequent. ■

**Definition 8.** A Record's Anomaly Score: The anomaly score of a maximal level  $n$  record is the sum of all its levels 1 to  $n$  frequent and infrequent itemsets' anomaly scores. ■



**Proposition 1.** A Normal/Anomalous Record Property: A normal record has more frequent than infrequent itemsets and has a negative total record anomaly score, while an anomalous record has more infrequent than frequent itemsets and has a positive or zero total record anomaly score. ■

### 3.3.2. The Proposed Apriori-Infrequent Algorithm

The goal of the Apriori-Infrequent Algorithm is to generate all frequent patterns as well as all infrequent patterns at every level, and be able to use this knowledge to compute anomaly scores for records. In order to compute frequent and non-frequent itemsets efficiently, the proposed algorithm argues that the Apriori's method for computing candidate  $(i+1)$ -itemsets by joining all frequent  $i$ -itemsets ( $L_i$ ) with themselves, if their first  $(i - 1)$  items are the same and the first itemset comes before the second itemset in the  $L_i$  list, can be improved on, with a third condition. The third join condition introduced by the Apriori-Infrequent algorithm states that an itemset in the  $L_i$  list will only be used to join other items in the  $L_i$  list that meet the first two conditions if this itemset's last item (or  $i^{\text{th}}$  item) appears in a joinable item list called Z list, consisting of all  $(i-1)$ th item of  $L_i$ . The purpose of the Z list is to prevent ahead of time, the need to join itemsets which produce itemset results that have no chance of being frequent because their subsets are not frequent. Such itemsets in the Apriori algorithm are pruned during this step but we avoid both creating them in the first place, computing their subsets and pruning them. Our algorithm looks for infrequent patterns (which were frequent in the previous level but when they are combined with some other attributes, they become infrequent). These infrequent itemsets are similar to negative borders [MT04], but is computed in a more efficient fashion in our online Apriori algorithm. This concept of fast detection of infrequent pattern is useful for intrusion detection domain because suppose for example, in connection record, Flag ACK (ACKnowledgement) is frequent but when ACK is combined with Flag SYN (SYNchronized), it may be an attack. The formal Apriori-Infrequent algorithm is given as Algorithm 3 [Figure 34] and the Smart-Join technique it uses is also given as Algorithm 2 [Figure 33].

**ALGORITHM 2.** (*Apriori-SmartJoin: Computing Candidate  $C_k$  from  $L_{k-1}$* )

**Algorithm Apriori-SmartJoin()**

**Input:** A list of large  $(k-1)$ -itemsets:  $L_{k-1}$ ,

**Output:** A list of candidate  $k$ -itemsets:  $C_k$ ,

**Other variables:** Z-list for smart join

**begin**

$C_k = \emptyset$

$Z =$  the set of all  $(k-2)$ th item in  $L_{k-1}$ .

for each pair of itemsets  $M$  and  $P \in L_{k-1}$  do

**begin**

$M$  joins with  $P$  to get itemset  $M \cup P$

if the following conditions are satisfied,

(a) itemset  $M$  comes before itemset  $P$  in  $L_{k-1}$

(b) the first  $k-2$  items in  $M$  and  $P$  (excluding just the last item) are the same.

(c) the last item (or  $(k-1)$ th item) of each itemset in  $L_{k-1}$  is joinable only if this item is in the Z list.

if  $M$  and  $P$  are joinable then

$C_k = C_k \cup M \cup P$

**end**

**end**

Figure 33: Algorithm: Apriori-SmartJoin

The process of Algorithm 2 is as follows:

Suppose, input to the algorithm is,  $L_2 = \{AB, AC, BD, BF, CD, CF\}$  and output of the algorithm would be  $C_3$ . So, in this example,  $k = 3$ .

At first step,  $C_3 = \emptyset$ . Then we create Z list,  $Z =$  set of  $3-2 = 1^{\text{st}}$  item in  $L_2$ . So,  $Z = \{A, B, C\}$ .

Then for each pair of itemset we do the following.

At first we take AB as M and AC as P. We check the following conditions:

1. AB comes before AC in  $L_2$ : true
2. First one item (A) is same at both itemsets: true

3. Last item of each itemset (B and C) is in Z list: *true*

Since all three conditions are true, AB and AC are joinable and we get

$$C_3 = \emptyset \cup AB \cup AC = \{ABC\}$$

Then we take AB as M and BD as P. We check the following conditions:

1. AB comes before AC in  $L_2$ : *true*
2. First one item (A) and (B) are same at both itemsets: *false*
3. Last item of each itemset (B and D) is in Z list: *false*

So, we will not join these two items and will proceed further. For the same reason, we cannot join the following pairs of items: (AB, BF), (AB, CD), (AB, CF), (AC, BD), (AC, BF), (AC, CD), (AC, CF), (BD, CD), (BD, CF), (BF, CD), (BF, CF). The following items cannot be joined because of only condition 3 is not satisfied: (BD, BF) and (CD, CF).

Following is the algorithm for Apriori-Infrequent.

**ALGORITHM 3.** (*Apriori-Infrequent: Computing Infrequent Patterns*)

**Algorithm Apriori-Infrequent()**

**Input:** *A list of candidate itemsets:  $C_1$ ,  
Minimum support count  $\lambda$*

**Output:** *A list of frequent itemsets:  $L$ ,  
Anomaly score of each record.*

**Other variables:** *A list of Infrequent itemsets:  $S$ ,*

**begin**

$k = 1$

1. *Compute frequent  $L_k$  and infrequent  $S_k$  with minimum support  $\lambda$  from  $C_k$ .*

2. *While ( $L_k \neq \emptyset$ ) do*

*begin*

2.1.  $k = k+1$

2.2. *Compute the next candidate set  $C_k$  from  $L_{k-1}$  as  $L_{k-1}$  Apriori-smart join  $L_{k-1}$*

2.3. *For each itemset in  $C_k$  do*

2.3.1. *Calculate all possible subsets and prune if not previously large*

2.4. *If  $C_k = \emptyset$  then break and go to step 3*

2.5. *Compute frequent  $L_k$  and infrequent  $S_k$  with minimum support  $\lambda$  from  $C_k$ .*

2.6. *Update anomaly score function with  $L_k$  and infrequent  $S_k$  (please refer to section 3.3.3 for details)*

*end*

3. *Compute all Frequent patterns as  $L = L_1 \cup \dots \cup L_k$*

**end**

Figure 34: Algorithm: Apriori-Infrequent

The process of the algorithm is explained in detail at Section 3.4 with an example application.

### 3.3.3. Anomaly Score Calculation

Given a record, an anomaly score is computed from all its level 1 to level  $n$  patterns (both frequent and non-frequent patterns), where  $n$  is the largest number of items in the maximal frequent pattern as presented in the definitions. To compute the anomaly score

of a record, each level  $k$  frequent pattern in the record is assigned an anomaly score of  $-k$ , while each level  $k$  infrequent pattern is assigned an anomaly score of  $+k$ , and the anomaly score of a record is the sum of the anomaly scores of all its frequent and infrequent patterns. If a record's total anomaly score becomes positive, then, this record has more infrequent than frequent patterns and is considered anomalous. On the other hand, if a record's anomaly score is negative, then, the record has more frequent than non-frequent patterns and is considered normal. If a record has zero anomaly score, it means it has the same number of frequent and infrequent patterns, and for increased security, the proposed system treats such a record as anomalous since it is safer to have a false alarm than harmful undetected intrusion. This anomaly detection module generates anomaly alerts for records with positive anomaly scores. The simple logic behind anomaly score weight assignment to frequent and infrequent itemsets is that the more the number of items in an infrequent itemset, the lower the chances of this itemset being in an arbitrary record. Thus, the presence of an infrequent 3-itemset is more rare than the presence of an infrequent 2-itemset in a record. Therefore, the anomaly weights of infrequent itemsets are proportionately increased with their size levels, while those of frequent itemsets are decreased with increasing number of items in the itemset. For example, while an infrequent 2-itemset like AC would have an anomaly score of  $+2$ , a frequent 2-itemset like AF would have anomaly score of  $-2$ . However, an infrequent 3-itemset would have an anomaly score of  $+3$ , while a frequent 3-itemset would have an anomaly score of  $-3$ .

The proposed WiFi Miner system is able to calculate or give each connection packet an anomaly score on the fly. This is an important step as it eliminates the need to generate association rules from frequent patterns as done by many existing approaches in order to identify intrusions. The simple anomaly score rule assigns a positive anomaly score of  $+n$  to every  $n$ -itemset infrequent pattern in a record that is equal to the number of items in the infrequent pattern but assigns a negative anomaly score of  $-n$  to a frequent pattern with  $n$  items. This rule is based on the premise that certain anomalies are infrequent events that embed themselves in frequent or normal packets. The anomaly score of each database transaction is computed in parallel with support counting of each level candidate set of the Apriori-Infrequent algorithm and this utilizes the records while they are still in

memory without incurring additional I/O costs. Thus, the total anomaly score of a record is computed as the sum of all the anomaly scores of this record's itemset level 1 to level  $n$  frequent and infrequent patterns, where  $n$  is the last nonempty level of frequent patterns for the record. A record is declared anomalous if its total anomaly score is zero or positive but normal if its total anomaly score is negative.

For Example, suppose at level two we have, Frequent-2-itemsets,  $L_2 = \{AB, AC, AD\}$  and Infrequent-2-itemsets,  $S_2 = \{BC, CD\}$ . Now, when we will be scanning the database for calculating the candidate-3-itemsets,  $C_3$  for the next level, we will check each transaction record for  $L_2$  and  $S_2$ . Suppose,  $i^{\text{th}}$  transaction,  $T_i = \{A, B, C, D\}$ . At this transaction we can see that it has frequent-2-itemsets AB, AC and AD. For each frequent itemset found we assign -2 and we can also see that it contains infrequent-2-itemsets BC and CD. For each infrequent itemset found we assign +2. So, the anomaly score of  $T_i$  at this level would be:  $-2 (AB) -2 (AC) -2 (AD) +2 (BC) +2 (CD) = -2$ . The final anomaly score of  $T_i$  would be the sum of all anomaly score calculated at each level. After getting final anomaly score of a transaction, if the score is Zero or Positive, we can say it as an anomalous record and if the score is negative, we can consider it as a frequent record.

### **3.4. Example Application of the Apriori-Infrequent and Anomaly Score**

Assume that wireless network connection records were captured and preprocessed to produce a database transaction table similar to columns one and two of Table 3, with candidate 1-items as  $\{A, B, C, D, E, F\}$ . In pre-processed wireless packets or records, the attributes depicted as A to F above would represent connection features like: connection date and time, source and Destination MAC address, packet size in bytes, access point MAC address (BSSID), Frame Type/Subtype, transmission rate, Client/AP sequence number, signal power, access point name, source type (station or access point), channel, etc.

TID	Items	Anomaly Score		
		Pass 1	Pass2	Final Score
1	A B D	$-3+0 = -3$	$-4+2 = -2$	-5
2	A C E F	$-2+2 = 0$	$-2+10 = +8$	+8
3	B C D F	$-3+1 = -2$	$-4+8 = +4$	+2
4	A B C D	$-4+0 = -4$	$-8+4 = -4$	-8
5	A B C E	$-3+1 = -2$	$-6+6 = 0$	-2

Table 24: Database records Anomaly Score

**Example 1:** Using the WiFi Miner Apriori-Infrequent and Anomaly score counting technique, identify the anomaly or alert records from Table 24 (first two columns) if the minimum support threshold is 60% or 3 out of 5 transactions.

**Solution 1:** Applying Algorithm 3 (figure 34),  $C_1 = \{A:4, B:4, C:4, D:3, E:2, F:2\}$ , and  $L_1 = \{A, B, C, D\}$  with anomaly score each of -1 and  $S_1 = \{E, F\}$  with anomaly score each of +1. The anomaly scores of the transactions in the database table are computed at this level as: TID 1, ABD has an anomaly score of  $-1(A) -1(B) -1(D) = -3$ . TID 2, ACEF has an anomaly score of  $-1(A) -1(C) +1(E) +1(F) = 0$ . The anomaly scores of transactions 3, 4 and 5 are respectively: -2, -4, and -2. Next, we compute  $C_2$  as  $L_1$  Apriori-gen join  $L_1$  since the Z list at this level is still empty set. Thus,  $C_2 = \{AB:3, AC:3, AD:2, BC:3, BD:3, CD:2\}$ .  $L_2$  is computed as  $\{AB, AC, BC, BD\}$  with anomaly score of -2 each, while  $S_2$  is computed as  $\{AD, CD\}$  with anomaly score of +2 each. The anomaly scores of the database transactions are updated as: Tid 1 (ABD) =  $-3(\text{score from previous step}) - 2(AB) + 2(AD) - 2(BD) = -5$ . Tid 2 (ACEF) =  $0(\text{score from previous step}) - 2(AC) + 2(AE) + 2(AF) + 2(CE) + 2(CF) + 2(EF) = +8$ . The rest of the anomaly scores are updated as shown in column 4 of table 24. During iteration 3, to create  $C_3$  list, the Z list is first created from  $L_2$  as item (2 -1) or the first item in each  $L_2$  itemset. Thus,  $Z = \{A, B\}$ . To join an  $L_2$  itemset, if the last element of the itemset is not in the Z list, then, we should not perform the join. This means that we first reduce our  $L_2 = \{AB, AC, AD, BC, BD,$

CD} to {AB} since AC, AD, BC, BD and CD do not have their last elements in the Z list. Thus, our  $C_3 = \{AB\}$  Apriori-gen join  $\{AB\} = \emptyset$ ; Since  $C_3 = \emptyset$ ; as well as  $L_3 = \emptyset$ , the algorithm ends without computing the anomaly score for this iteration. All records with negative anomaly scores are normal while those with positive or zero anomaly scores are alerts. The final anomaly scores of the example connection records are as given in Column 5 of Table 24.



## 4. EXPERIMENTS AND PERFORMANCE ANALYSIS

We developed WiFi Miner with JAVA language under Windows platform. This prototype system consists of all modules and algorithms described in Chapter 3. We used hardware sensor to capture wireless connection records before they reach the access point, then these captured records are preprocessed by Commview for WiFi software, which outputs the csv file. At last Anomaly Detection Module used this csv file to flag anomalous connection records.

The main objective of this experiment is to prove that WiFi Miner is capable of detecting more kinds of wireless attacks at a lower cost. We compared our system with Snort-Wireless [Loc05], which is the only open source wireless IDS and ADAM [BCJ+01], which uses the Apriori and association rule algorithm. Due to the unavailability of labeled wireless data, we crafted our own packets to test the system.

The rest of the chapter is organized as follows: section 4.1 describes our experimental setup, section 4.2 describes how we crafted the attack packets of different types and section 4.3 describes the test results and performance evaluation of our system.

### 4.1. *Experimental Setup*

Our testing environment consists of three computers (PC1, PC2 and PC3), one access point (AP1) and one wireless sensor (Network Chemistry sensor). We installed Network Chemistry sensor and Commview for WiFi in PC1 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 PC2 and PC3 where both were connected to AP1. PC2 is the attacker PC and PC3 is the victim PC. The topology of the network setup is depicted in figure 35.

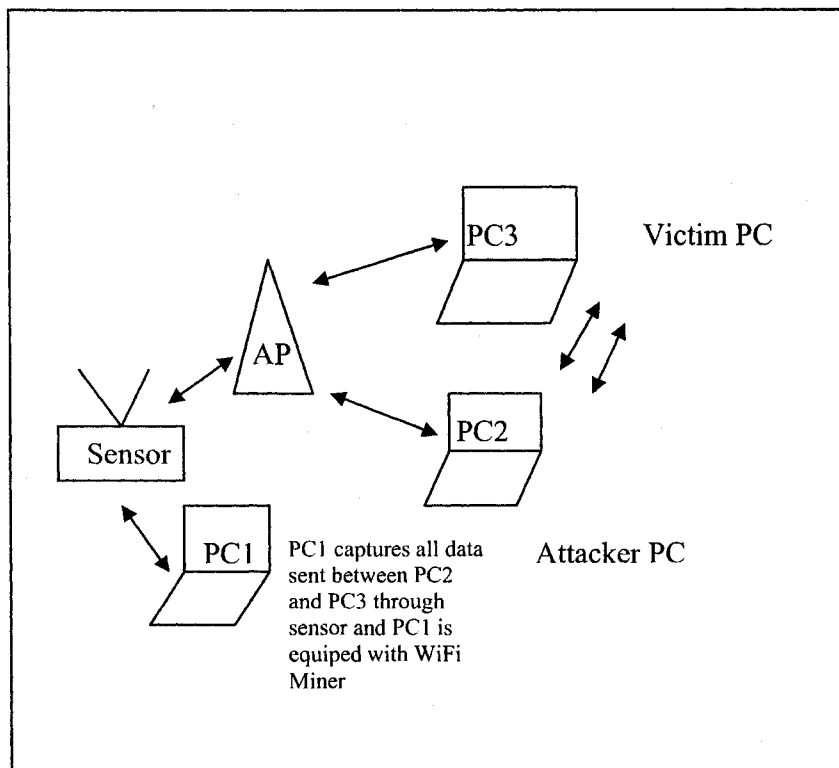


Figure 35: Experimental Network Setup

The hardware configurations of these PCs are as below:

PC1: Intel Centrino 1.50 GHz, 512 MB Ram, 60 GB Hard Drive

PC2: Intel Pentium 4, 3.07 GHz, 1.0 GB Ram, 149 GB Hard Drive

PC3: Intel Pentium 4 1.50 GHz, 512 MB Ram, 120 GB Hard Drive

#### 4.2. Attacks Used in Experiment

As described in section 1.4.7, wireless intrusions can be classified into four groups namely: Passive Attacks, Active Attacks, Man-in-the-Middle Attacks and Jamming Attacks. Since Jamming Attacks are not very common in nature because of the expense of acquiring hardware capable of launching jamming attacks, we will not consider this

kind of attacks in our experiments. To ensure that our system is capable of detecting wireless intrusions, we crafted attacks of the following types:

1. Passive Attacks (WEP Crack attacks and Port scanning attacks)
2. Active Attacks (SYN Flood attacks and UDP Flood attacks) and
3. Man-In-The-Middle Attacks (Rogue AP attacks)

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

### **Passive Attacks**

To test our system with WEP Cracking attack, we have used BackTrack network security suite [BT07]. BackTrack is a Linux distribution distributed as a Live CD and includes over 300 security tools that can be used in crafting attacks. Any Windows OS PC can be booted into linux mode to use these security tools with BackTrack CD.

At first we captured some packets from our Access Point (WiFiMiner) and from there we spoofed a valid client's MAC address. Then we started BackTrack security tool and using the Aireplay [Air07a] utility we sent authentication and association request to WiFiMiner AP (figure 36). The command was:

```
Aireplay-ng -l 0 -e WiFiMiner -a 00:14:D1:3A:71:E4 -h 00:0E:35:07:A7:FC eth0
```

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

Then we started sending fake ARP packets to WiFiMiner AP so that we can capture the replies through Airodump (figure 37). Once we have captured enough packets then we started Aircrack utility of decrypt the WEP key (Figure 38). Figure 39 shows the processing of data to find the WEP key. Within 15 minutes time frame Aircrack decrypted the WEP key (Figure 40).

```

Shell - Airodump
Shell
-x nbpps : number of packets per second
-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

stax ~ # aireplay-ng -l 0 -e WiFiMiner -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

stax ~ # aireplay-ng -3 -b 00:14:D1:3A:71:E4 -h 00:0E:35:D7:A7:FC -x 600 eth0
Saving ARP requests in replay_arp-0718-003643.cap
You should also start airodump-ng to capture replies.

```

Figure 36: Sending Fake Authentication packets

```

Shell - Airodump
Shell
-x nbpps : number of packets per second
-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

stax ~ # aireplay-ng -l 0 -e WiFiMiner -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

stax ~ # aireplay-ng -3 -b 00:14:D1:3A:71:E4 -h 00:0E:35:D7:A7:FC -x 600 eth0
Saving ARP requests in replay_arp 0718 003643.cap
You should also start airodump-ng to capture replies.

```

Figure 37: Saving Fake ARP packets

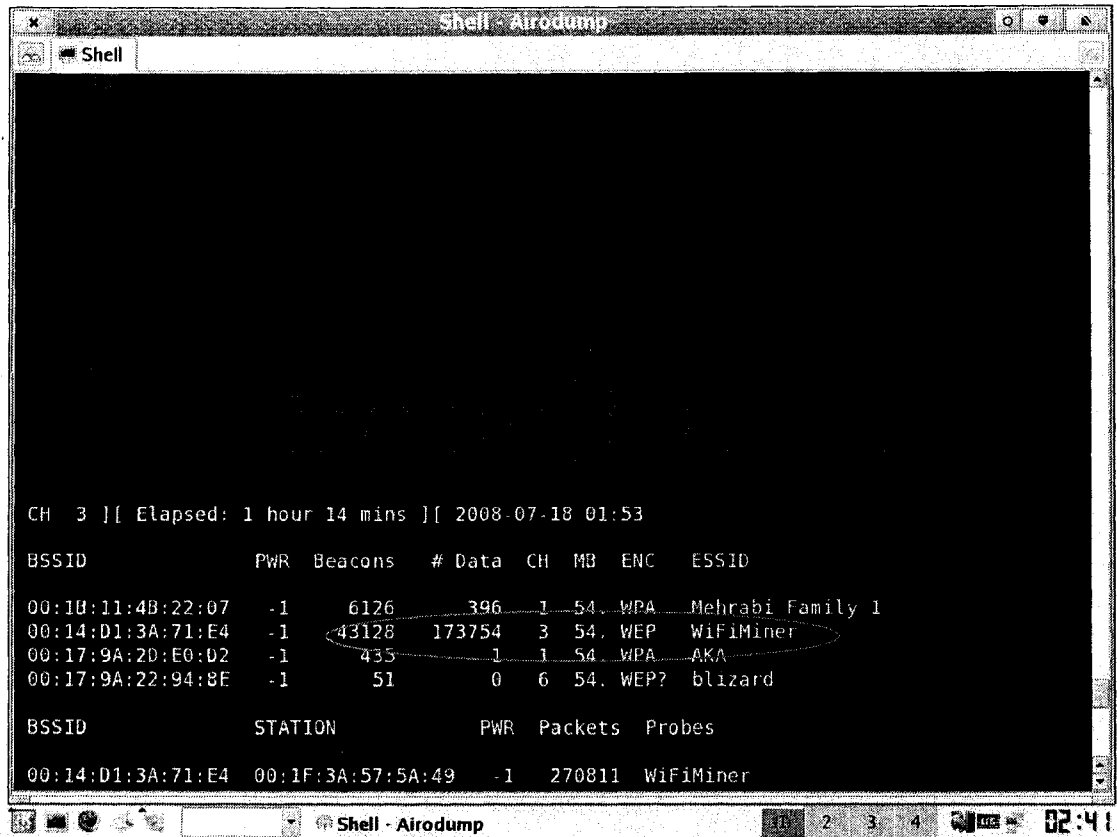


Figure 38: Gathering enough packet for WiFiMiner AP to decrypt WEP key

Our sensor captured all these fake ARP packets and we preprocessed these packets with Commview software and it generated a csv file containing these attack packets (figure 41).



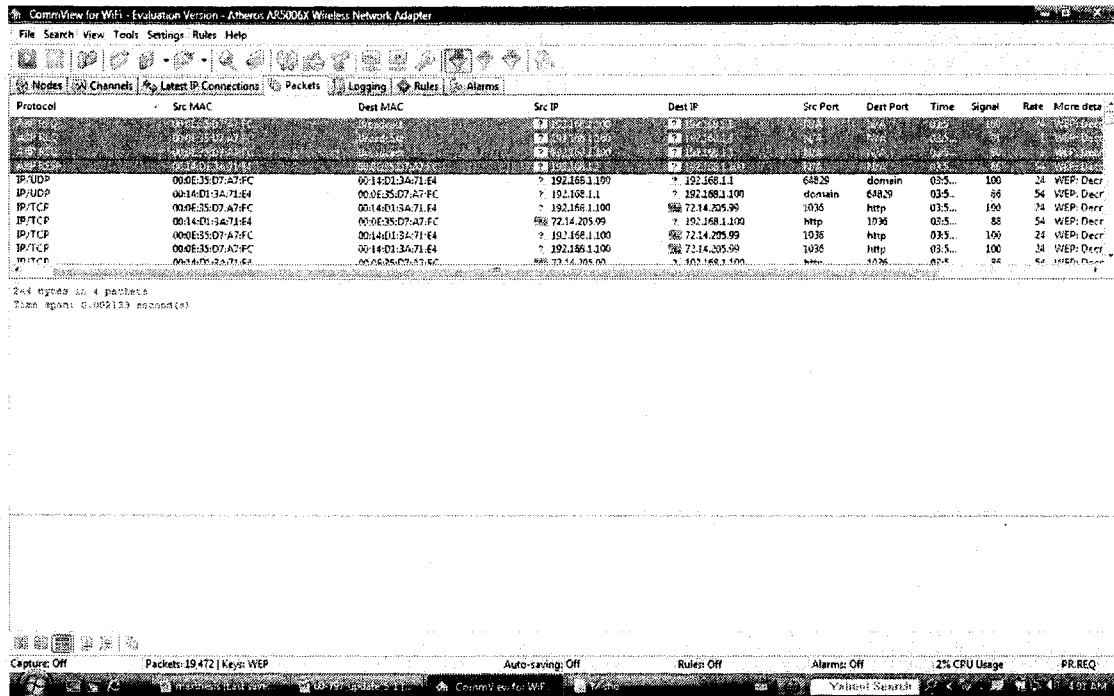


Figure 41: Fake ARP packets captured and preprocessed by WiFiMiner

The log of these crafted passive attack packets can be found at [http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/passive\\_attack\\_log.csv](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/passive_attack_log.csv).

### Active Attacks

To test active attacks, we have used Engage Packet Builder [Eng07] software to craft attack packets for SYN Flood and UDP Flood attacks. The signature we used to craft these attack packets were as follows:

SYN Flood: *flag = SYN, dest-host = victim (same), dest-service = vulnerable port (same)*  
 UDP Flood: *dst-host = victim (same), dst-service = vulnerable port/random port*

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 (192.168.1.199) at the place of source IP address and at destination IP address we put the victim's IP address (192.168.1.101). At source port we put some arbitrary port number (100) and destination port is some vulnerable port (80). 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 42. 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 43.



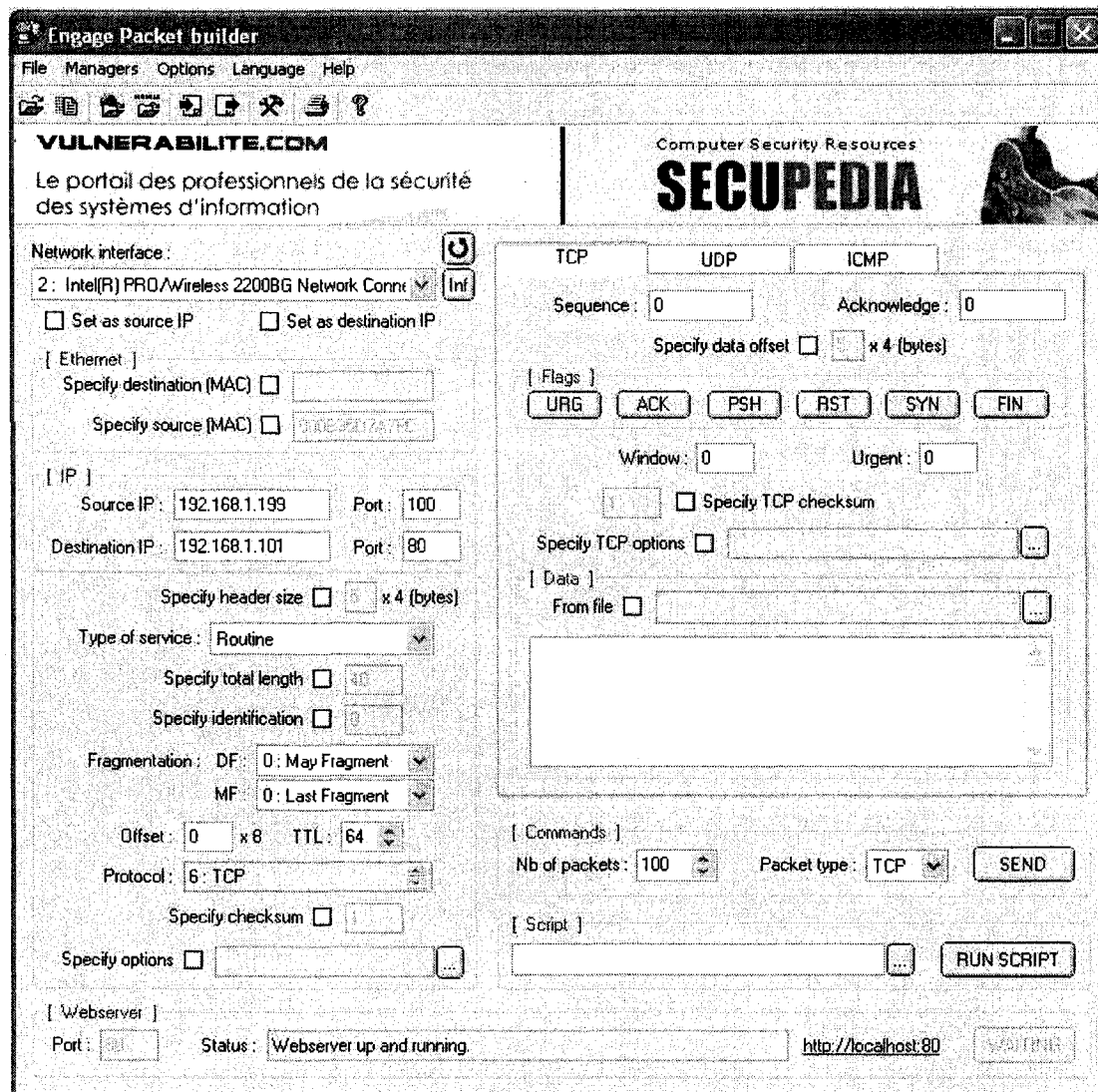


Figure 42: Interface for creating TCP Flood packets

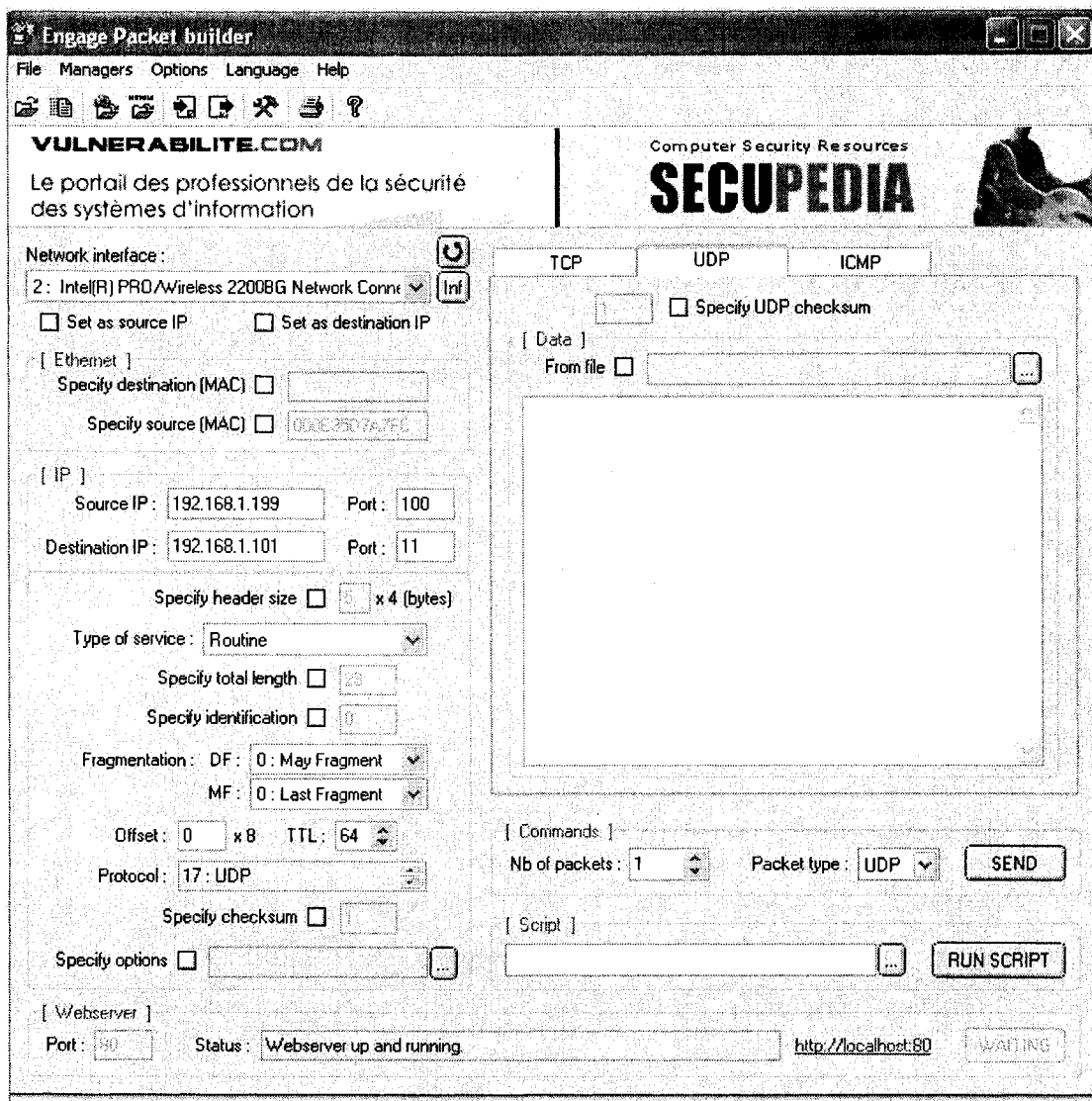


Figure 43: Interface for creating UDP Flood Attack packets

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. Figure 44 shows that we have successfully captured and preprocessed SYN Flood attack packets with Commview for WiFi and Figure 45 shows that of UDP Flood Attack packets.

# WiFi Miner: An Online Apriori and Sensor Based Wireless Network Intrusion Detection System

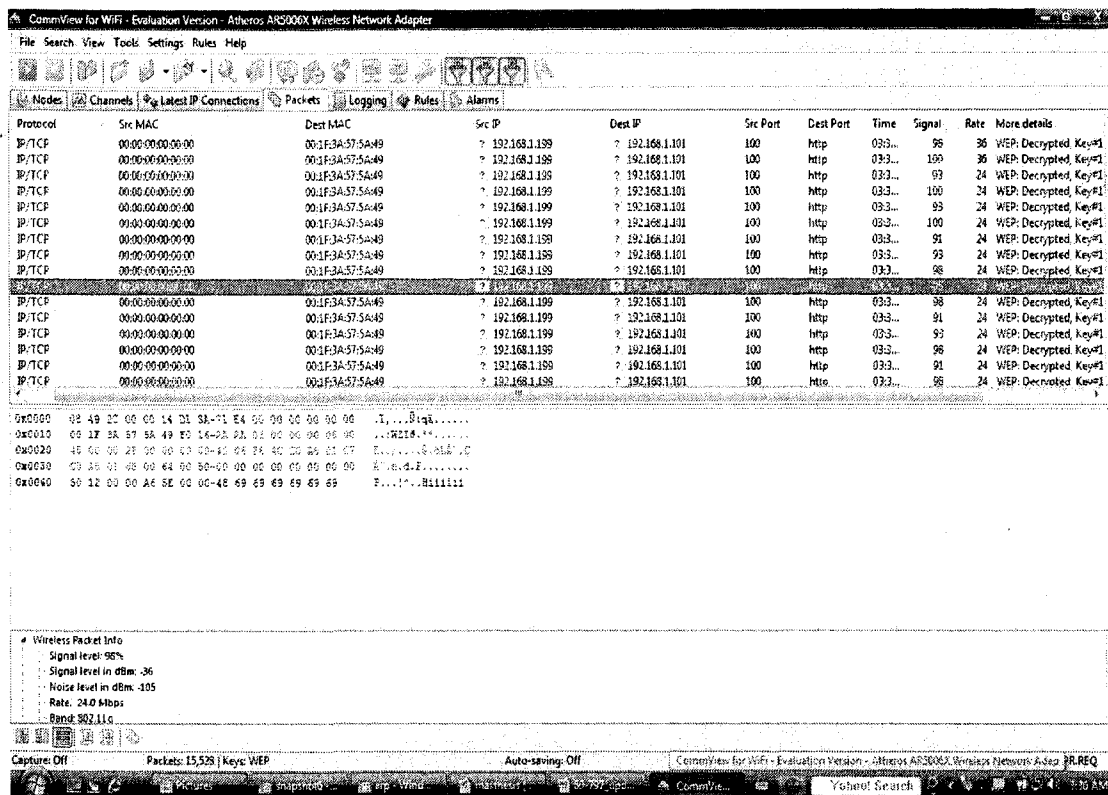


Figure 44: SYN Flood Attack packets captured by Commview

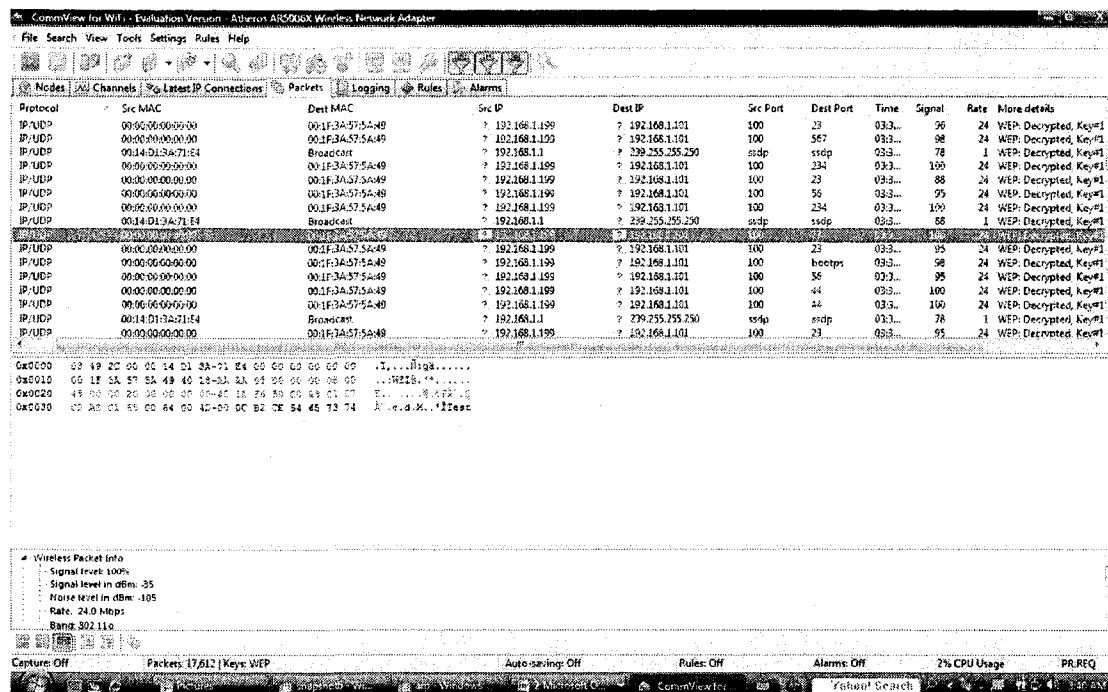


Figure 45: UDP Flood Attack packets captured by Commview

The log of these crafted active attack packets can be found at [http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/active\\_attack\\_log.csv](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/active_attack_log.csv).

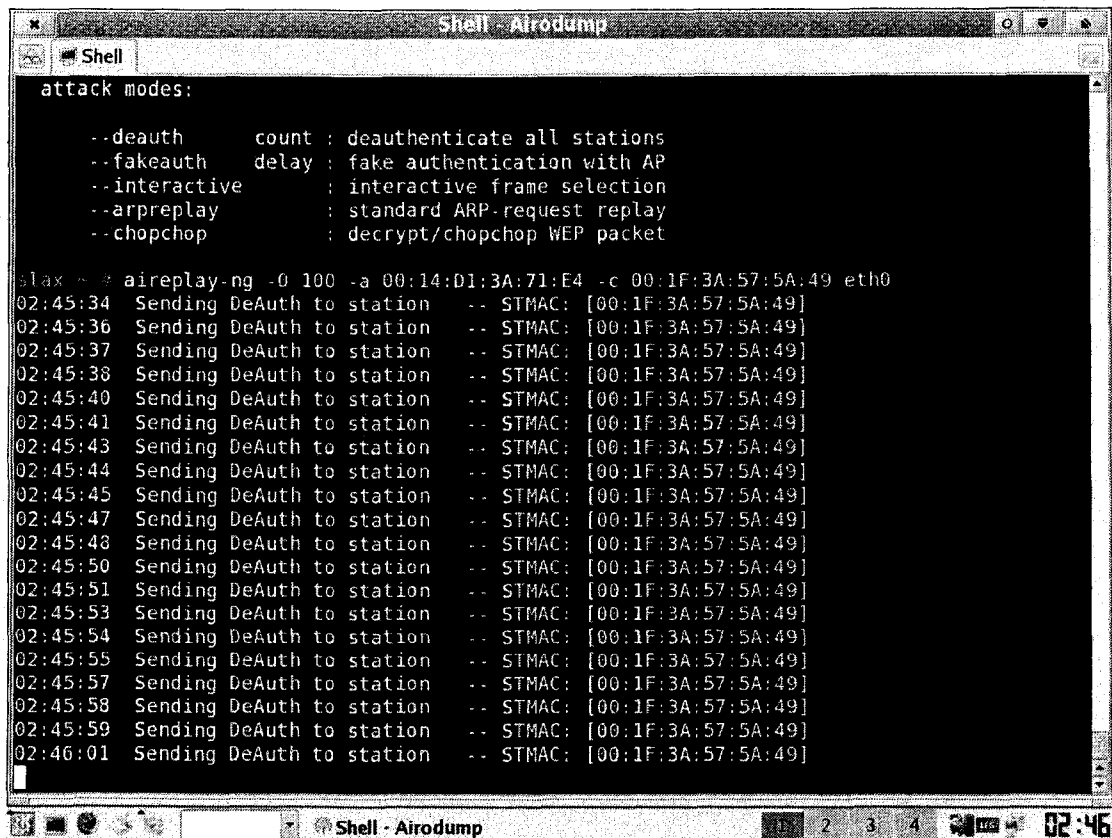
### **Man-In-The-Middle Attacks**

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 WiFiMiner (MAC address: 00-14-D1-3A-71-E4) operating at channel 3 and the victim's MAC address was 00-1F-3A-57-5A-49, which was connected to the legitimate AP. We placed another router with the same SSID (WiFiMiner) 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 46:

```
aireplay-ng -0 100 -a 00:14:D1:3A:71:E4 -c 00:1F:3A:57:5A:49 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 00:1F:3A:57:5A:49 is victim's MAC address and ath0 is the network card in use. This step is shown in Figure 45. 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.



```
Shell - Airodump
attack modes:
--deauth      count : deauthenticate all stations
--fakeauth    delay : fake authentication with AP
--interactive  : interactive frame selection
--arpreply    : standard ARP-request replay
--chopchop    : decrypt/chopchop WEP packet

slax ~ # aireplay-ng -0 100 -a 00:14:D1:3A:71:E4 -c 00:1F:3A:57:5A:49 eth0
02:45:34 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:36 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:37 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:38 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:40 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:41 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:43 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:44 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:45 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:47 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:48 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:50 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:51 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:53 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:54 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:55 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:57 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:58 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:45:59 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
02:46:01 Sending DeAuth to station -- STMAC: [00:1F:3A:57:5A:49]
```

Figure 46: Sending Deauthentication packets to the legitimate AP

These de-authentication packets were captured and gathered as anomalous packets with Commview for WiFi software as shown in figure 47.

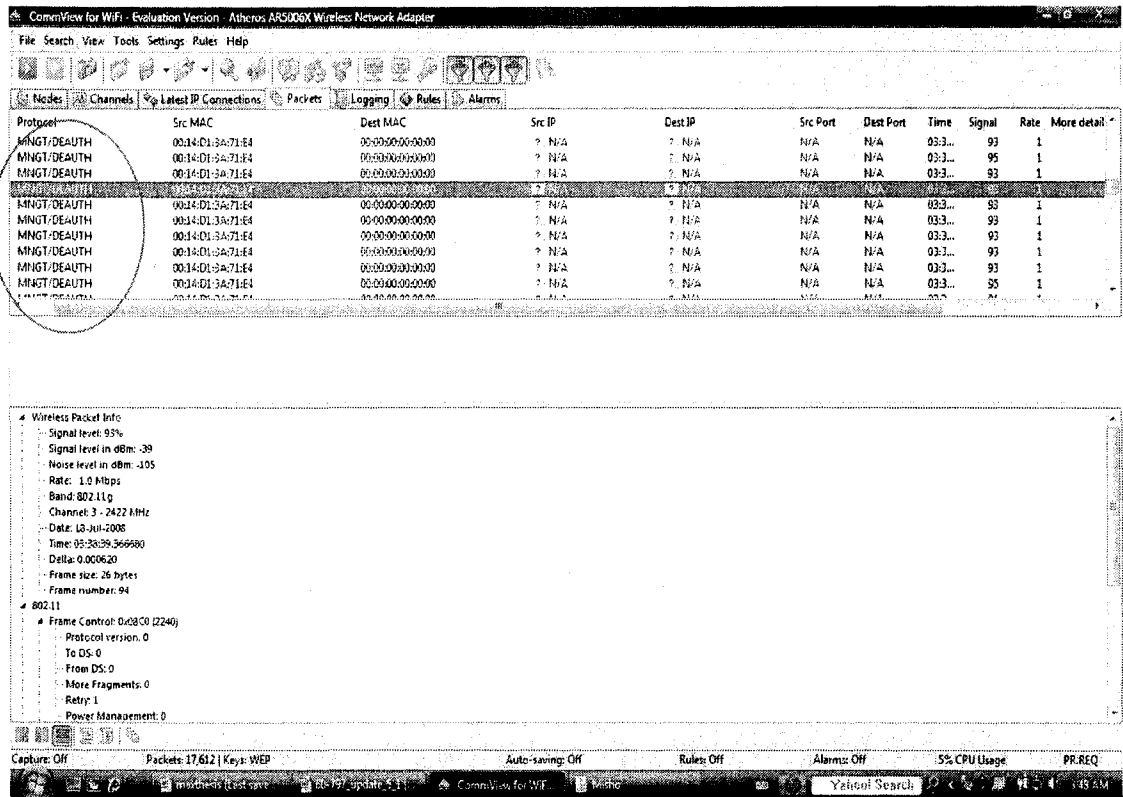


Figure 47: Deauthentication packets captured by Commview

The log of these crafted Man-In-The-Middle attack packets can be found at [http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/mim\\_attack\\_log.csv](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/mim_attack_log.csv).

### 4.3. Test Result and Performance Evaluation

At first, we have compared the runtime of our algorithm: Real-time Online Apriori-Infrequent Algorithm with traditional Apriori algorithm concept used in ADAM and noticed an around 35% increase in execution time efficiency in our algorithm as shown in Figure 48. This is because we are not generating association rules with confidence value and also we have improved the join and prune sections of the algorithm with our Smart-Join approach. It should be stated here that from analysis and experiments, the proposed Online Apriori with smart join produces complete and correct frequent and infrequent patterns as the regular Apriori algorithm given the same datasets.

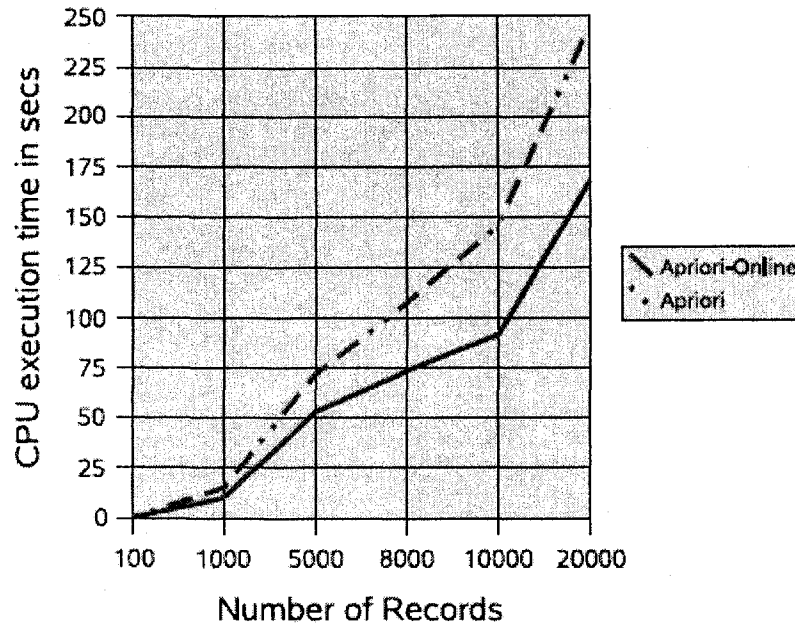


Figure 48: Comparison of Apriori-Infrequent with Apriori

Then, to test the system with these anomalous packets crafted at Section 4.2, at first we have generated some innocent packets between PC2 and PC3 in figure 35. These packets were generated as a result of some innocent web browsing. Within 5 minutes time window we have captured around 19,500 packets. Figure 49 shows that we captured these packets with the Network Chemistry sensor (log of these innocent packets can be found at [http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/innocent\\_log.cap](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/innocent_log.cap)). Then we preprocessed these data with Commview for WiFi software (log of these preprocessed data can also be found at [http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/innocent\\_log.csv](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/innocent_log.csv)) (figure 50).

# WiFi Miner: An Online Apriori and Sensor Based Wireless Network Intrusion Detection System

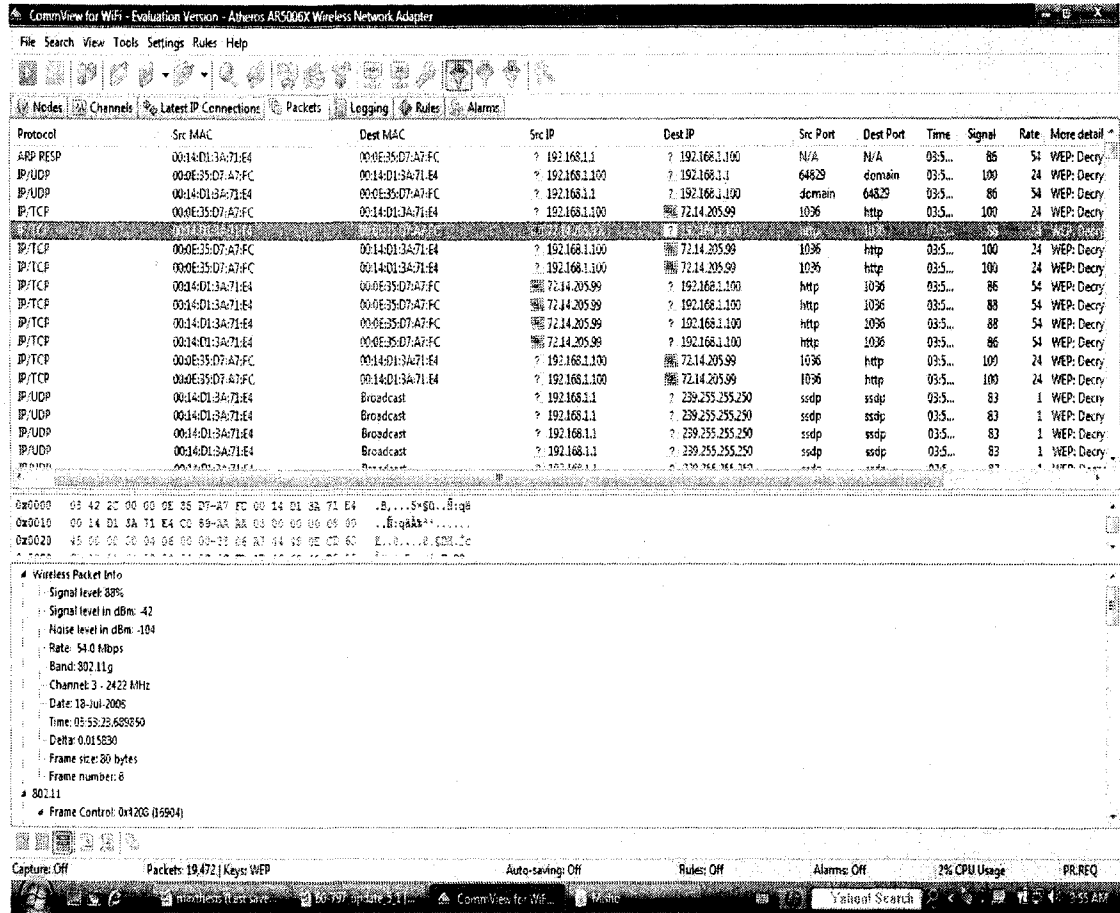


Figure 49: Innocent packets collected to test the system along with anomalous packets (complete log can be found @ [http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/innocent\\_log.cap](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/innocent_log.cap))



No	Protocol	Src MAC	Dest MAC	Src IP	Dest IP	Src Port	Dest Port	Time	Size	Signal	Rate	More details
1	ARP REQ	00:0E:35:C0:00:00	Broadcast	192.168.1.1	192.168.1.1	N/A	N/A	53:23.6	60	100		24 WEP: Decrypted, Key#1
2	ARP REQ	00:0E:35:C0:00:00	Broadcast	192.168.1.1	192.168.1.1	N/A	N/A	53:23.6	60	91		1 WEP: Decrypted, Key#1
3	ARP REQ	00:0E:35:C0:00:00	Broadcast	192.168.1.1	192.168.1.1	N/A	N/A	53:23.6	60	90		1 WEP: Decrypted, Key#1
4	ARP RESP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	192.168.1.1	N/A	N/A	53:23.6	64	86		54 WEP: Decrypted, Key#1
5	IP/UDP	00:0E:35:C0:14:D1	:192.168.1.1	192.168.1.1	192.168.1.1	64829	domain	53:23.6	91	100		24 WEP: Decrypted, Key#1
6	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	192.168.1.1	domain	64829	53:23.7	427	86		54 WEP: Decrypted, Key#1
7	IP/TCP	00:0E:35:C0:14:D1	:192.168.1.1	192.168.1.1	72.14.205	1036	http	53:23.7	80	100		24 WEP: Decrypted, Key#1
8	IP/TCP	00:14:D1:30:0E:35	C:72.14.205	192.168.1.1	192.168.1.1	http	1036	53:23.7	80	88		54 WEP: Decrypted, Key#1
9	IP/TCP	00:0E:35:C0:14:D1	:192.168.1.1	192.168.1.1	72.14.205	1036	http	53:23.7	72	100		24 WEP: Decrypted, Key#1
10	IP/TCP	00:0E:35:C0:14:D1	:192.168.1.1	192.168.1.1	72.14.205	1036	http	53:23.7	555	100		24 WEP: Decrypted, Key#1
11	IP/TCP	00:14:D1:30:0E:35	C:72.14.205	192.168.1.1	192.168.1.1	http	1036	53:23.7	72	86		54 WEP: Decrypted, Key#1
12	IP/TCP	00:14:D1:30:0E:35	C:72.14.205	192.168.1.1	192.168.1.1	http	1036	53:23.7	1502	88		54 WEP: Decrypted, Key#1
13	IP/TCP	00:14:D1:30:0E:35	C:72.14.205	192.168.1.1	192.168.1.1	http	1036	53:23.7	1502	88		54 WEP: Decrypted, Key#1
14	IP/TCP	00:14:D1:30:0E:35	C:72.14.205	192.168.1.1	192.168.1.1	http	1036	53:23.7	332	86		54 WEP: Decrypted, Key#1
15	IP/TCP	00:0E:35:C0:14:D1	:192.168.1.1	192.168.1.1	72.14.205	1036	http	53:23.7	72	100		24 WEP: Decrypted, Key#1
16	IP/TCP	00:0E:35:C0:14:D1	:192.168.1.1	192.168.1.1	72.14.205	1036	http	53:23.9	72	100		24 WEP: Decrypted, Key#1
17	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:25.9	312	83		1 WEP: Decrypted, Key#1
18	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:25.9	330	83		1 WEP: Decrypted, Key#1
19	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:25.9	384	83		1 WEP: Decrypted, Key#1
20	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:25.9	376	83		1 WEP: Decrypted, Key#1
21	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:25.9	306	83		1 WEP: Decrypted, Key#1
22	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:25.9	348	81		1 WEP: Decrypted, Key#1
23	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:26.0	380	83		1 WEP: Decrypted, Key#1
24	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:26.0	326	81		1 WEP: Decrypted, Key#1
25	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:26.0	378	83		1 WEP: Decrypted, Key#1
26	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:26.0	372	83		1 WEP: Decrypted, Key#1
27	IP/UDP	00:14:D1:30:0E:35	C:192.168.1.1	192.168.1.1	239.255.255.255	ssdp	ssdp	53:26.0	372	83		1 WEP: Decrypted, Key#1

Figure 50: Preprocessed innocent packets (A complete log can be found @ [http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/innocent\\_log.csv](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/innocent_log.csv))

After that we collected 500 anomalous packets (200 packets from passive attacks, 200 packets from active attacks and 100 packets from Man-in-the-middle attacks) and mixed these anomalous packets with innocent packets and input the total combined dataset into our Anomaly Detection Module where we run the Apriori-Infrequent algorithm. A complete log of these crafted attack packets can be found at [http://cs.uwindsor.ca/~woddlab/sensor\\_mine/wifi\\_miner/attack\\_log.csv](http://cs.uwindsor.ca/~woddlab/sensor_mine/wifi_miner/attack_log.csv).

The algorithm outputs some packets as alert which have positive anomaly score, and then we check these alert records if they really belong to the group of 500 anomaly packets to calculate the anomaly detection rate and false alarm rate. We tested the same dataset with

SNORT Wireless and traditional Apriori based system ADAM to compare our system with these two existing system.

Detected	(Out of 500 Attacks)		
	WiFi Miner	SNORT Wireless	ADAM
Attacks Detected	433	335	377
False Alarm	180	292	248

Table 25: Attacks Detected and False Alarm Comparison

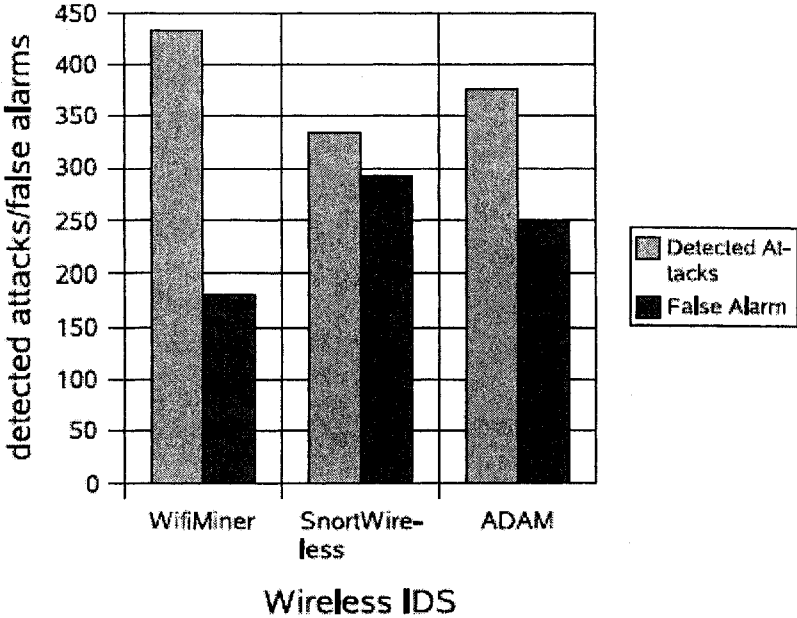


Figure 51: Comparison of Attacks detected and false alarm rate

Detected	3 Algorithms		
	WiFi Miner	SNORT Wireless	ADAM
Passive Attacks (200 Attacks)	179 (89.5%)	138 (69%)	161 (80.5%)
Active Attacks (200 Attacks)	171 (85.5%)	145 (72.5%)	151 (75.5%)
Man-In-The-Middle Attacks (100 Attacks)	83 (83%)	52 (52%)	65 (65%)

Table 26: Specific Attacks Detection Comparison

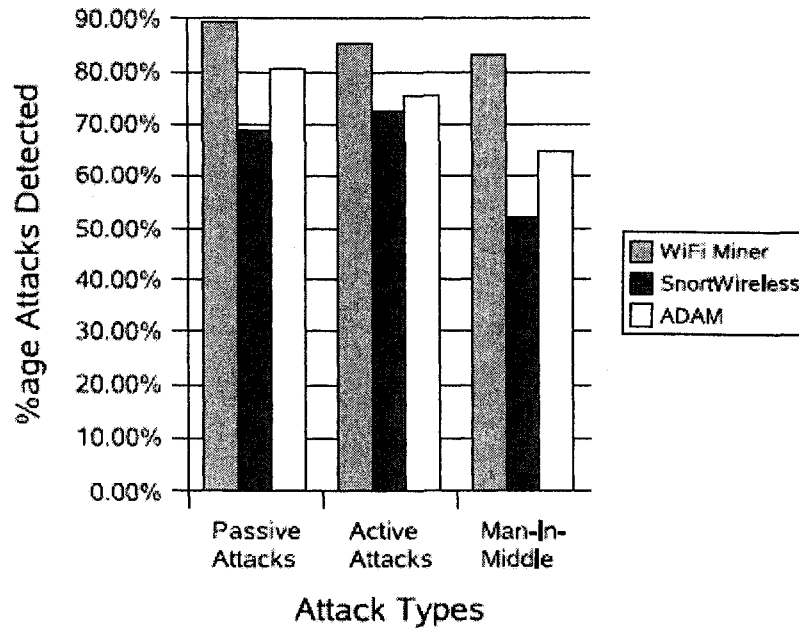


Figure 52 Specific attacks comparisons of WiFi Miner with SNORT Wireless and ADAM

**Result Analysis:**

From Table 25, we can see that our proposed system, WiFi Miner, performed better than SNORT Wireless and ADAM and false alarm rate is also reduced. The experiment also shows that without any training data our system can perform well and can detect intrusive packets efficiently.

Table 26, gives a detailed view of specific attack detection rate. From there we can see that WiFi Miner performed better at detecting passive attacks (detection rate is 89.5%) than detecting active attacks (85.5%). In case of Man-In-The-Middle attack, WiFi Miner detected 83% of attacks which is lower than detection rate of other two types of attacks but still in comparison to SNORT Wireless and ADAM, it performed better.

Currently, the proposed WiFi Miner system has no mechanism for detecting Jamming Wireless attacks. Also, if the minimum support is set too low, there may be large number of frequent itemsets and fewer infrequent itemsets. As a result, attacks may go undetected. Experiments show that for this wireless intrusion detection domain, a good choice of minimum support is 60% or more. Future work should explore improving efficiency of the system, handling more types of attacks and further reduction of false alarms.

## 5. CONCLUSION AND FUTURE WORKS

This paper proposes a wireless intrusion detection system: WiFi Miner, which uses Apriori-Infrequent based algorithm to detect infrequent patterns, then our algorithm designed for Anomaly Score Calculation, assigns a score to each wireless packet. Positive or zero anomaly score in a specific connection record means that more infrequent/anomalous patterns are found in that record than frequent patterns while a negative anomaly score indicates a normal packet. We have also used proprietary Network Chemistry hardware sensors to capture real-time traffic in order to improve intrusion response time.

Our system is different from existing wireless intrusion systems, since it eliminates the need for hard-to-get training data and detects intrusions in real time. Also, like other existing wireless intrusion systems, it captures the packets from airwaves while wired IDSs use net-flow data from routers. Thus, the major contribution of our system is that it can detect anomalous packets in real time without any training phase. We have tested our system with crafted intrusions and compared it with other two systems and found our system to be more efficient. Another major contribution is that we have introduced Smart-Join, which is an improved version of Join and Pruning steps in original Apriori algorithm.

The only critical step in this system is choosing the right support rate. Because, if the support rate is not chosen correctly or if it is set too low, then the false alarm rate will increase significantly as normal and innocent packets will also be flagged as alerts. Based on our study, we recommend minimum support rate 60% to be suitable for this system. Right now our system is not capable of detecting any kind of Jamming Attacks.

In the future, we plan to enhance our system to work with many access points, currently it is capable of handling wireless connection records from one access point although our sensors are capable of finding all APs in their ranges. We are also working towards making our system generalized so that it can be used for both wired and wireless

intrusion detection. Other future works include applying this online intrusion detection system approach to other domains like environment pollution monitoring systems where excessive levels of pollution can quickly raise alerts as anomalies from sensor captured data.

## Bibliography:

1. [AJ01] R Agarwal, MV Joshi; PNrul: A New Framework for Learning Classifier Models in Data Mining; *on First SIAM International Conference on Data Mining*, pp. 1 – 17, April 5-7, 2001;
2. [Air08a] Airmagnet Wireless Network Assurance. [www.airmagnet.com](http://www.airmagnet.com). 2008.
3. [Air08b] AirDefense Anywhere Anytime Wireless Protection. [ww.airdefense.net](http://ww.airdefense.net). 2008.
4. [Air08c] AirWave – Wireless Network Management Software WiFi, Mesh and more. [www.airwave.com](http://www.airwave.com). 2008.
5. [Air07a] Aircrack. Airdump webpage. <http://airdump.net/papers/packet-injection-windows>, 2007
6. [Air07b] Wild Packets Illuminate Your Network. <http://www.wildpackets.com/>. 2007.
7. [Air07c] Air Snort Homepage. <http://airsnort.shmoo.com/>. 2007.
8. [Ba00] Tim Bass. “Intrusion Detection System and Multi-sensor Data Fusion”. *Communication of the ACM*, pp 99-105, vol. 43, no. 4, April 2000
9. [Ba03] Yuebin Bai. “Intrusion Detection Systems: Technology and Development”. *Advanced Information Networking and Applications, 2003. AINA 2003. 17<sup>th</sup> International Conference on, 27-29 March 2003*.
10. [Be03] Berkhin, Pavel. “Survey of clustering Data Mining Techniques”. *A whitepaper published from Accrue Software Inc. 2003*.
11. [Blu08] BlueSocket – Enterprise Wireless LAN security and Management Solutions. <http://www.bluesocket.com/>. 2008.
12. [BBH+02] Jerzy Bala, Sung Balik, Ali Hadjarian, BK Gogia and Chris Manthorne. “Application of a Distributed Data Mining Approach to Network Intrusion Detection”. *Proceeding of the first international joint conference on Autonomous and multiagent Systems: Part 3, July 2002, pp 1419 – 1420*.

13. [BCH+01] Eric Bloedorn, Alan D. Chritiausen, W. Hill, C. Skorupka, Lisa M. Talbot, Jonathan Tivel. "Data Mining for Network Intrusion Detection: How to get started". *Technical report of the MITRE Corporation, VA*
14. [BCJ+01] Daniel Barbara, Julia Couto, Sushil Jadodia, Ningning Wu. "ADAM: A Testbed 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. Pages 15 – 24, 2001.*
15. [BCL02] D. Barbara, J. Couto, Y. Li. "Coolcat: An entropy based algorithm for categorical clustering". *In proceeding of the 11<sup>th</sup> ACM conference on Information and knowledge management (CIKM). Pages 582 – 589. McLean, VA, November 2002.*
16. [BCV01] E. Biermann, E. Cloete, L.M. Venter. "A comparison of Intrusion Detection System". *Computer and Security, Volume 20, Issue 8, pp. 676-683, December 1, 2001.*
17. [BLC03] Daniel Barbara, Yi Li, Julia Couto, Jia-Ling Lin, Sushil Jajodia. "Bootstrapping a data mining intrusion detection system". *Proceedings of the 2003 ACM symposium on applied computing, pp 421 – 425, Florida, USA, March 2003.*
18. [BK03] Bai, Y. and Kobayashi, H. Intrusion Detection Systems: Technology and Development. *Proceedings of the 17th International Conference on Advanced Information Networking and Applications. pp. 710-715. 2003.*
19. [BKN+00] Markus Breunig, Hans-Peter Kriegel, Raymond T. Ng and Jorg Sander."LOF: Identifying densitybased local outliers". *In proceedings of the ACM SIGMOD Conference, pp 93 – 104, Dallas, TX, 2000.*
20. [BR01] B. Balajinath, S.V. Raghavan. "Intrusion Detection through Learning Behavior Model". *Computer Communications, Volume 24, Issue 12, pp. 1202-1212, July 15, 2001.*
21. [BS03] J. Belardo, and S. Savage. "802.11 Denial-of-Service Attacks: real Vulnerabilities and Practical Solutions". *In proceedings of 12th USENIX Security Symposium, pp. 2 – 2, Washington DC, August 2003.*
22. [BT07] BackTrack Security Tools. <http://www.remote-exploit.org/backtrack.html>. 2007.



23. [BTS+01] Bloedorn, E.L. Talbot, C. Skorupka, A. Christiansen, W. Hill, and J. Tivel [2001]. "Data Mining applied to Intrusion Detection: MITRE experiences". *Submitted to the 2001 IEEE International Conference on Data Mining*.
24. [CG00] Chris Clifton, Gary Gengo. "Developing Custom Intrusion Detection Filters Using Data Mining". *MILCOM 2000. 21<sup>st</sup> Century Military Communications Conference Proceedings, Volume: 1, 22-25 Oct. 2000. Pages: 440 – 443 Vol. 1*
25. [CV07] CommView for WiFi – Wireless Network Analyzer and Monitor. <http://www.tamos.com/products/commwifi/>. 2007
26. [Du03] Margaret H. Dunham; Data Mining Introductory and Advanced Topics; Prentice Hall 2003, ISBN 0-13-088892-3
27. [De87] Denning, D. "An Intrusion Detection Model". *IEEE Transactions of software engineering SE – 13, 2. (Feb 1987), pp. 222 – 232*.
28. [DFP+04] Holger Dreger, Anja Feldmann, Vern Paxson, Robin Sommer. "Operational experiences with high volume network intrusion detection". *Proceedings of the 11th ACM conference on Computer and communications security, pp. 2 – 11, October 2004*.
29. [DH06] G. Deckerd, D.A. Hindarto. "Wireless Attacks from an Intrusion Detection Perspective". *A whitepaper on SANS Institute as part of Information Security Reading Room. pp. 1 – 25, November 23, 2006*.
30. [Eng07] Engage Security. Engage Security webpage. <http://www.engagesecurity.com>. 2007.
31. [EDA08] C.I. Ezeife, Jingyu Dong, A.K. Aggarwal, "SensorWebIDS: A Web Mining Intrusion Detection System", *In Vol 4, issue 1, 2008, of the International Journal of Web Information Systems(IJWIS), Emerald Group Publishing Limited*
32. [ELK+04] L Ertoz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar and P. Dokas, The MINDS - Minnesota Intrusion Detection System; *Book chapter in Data Mining: Next Generation Challenges and Future Directions 2004*.
33. [EMZ+00] Eleazar Eskin, Matthew Miller, Zhi-Da Zhong, George Yi, Wei-Ang Lee, Sal Stolfo. "Adaptive Model Generation for Intrusion Detection Systems".

*Workshop on Intrusion Detection and Prevention, 7th ACM Conference on Computer Security, Athens, GR: November, 2000*

34. [FSL+01] Wei Fan, Salvatore J. Stolfo, Wenke Lee, Matthew Miller, Philip K. Chan. "Using Artificial Anomalies to Detect Unknown and Known Network Intrusions". *Data Mining 2001, ICDM 2001, proceedings IEEE International Conference on*, pp. 507 – 527, 29 Nov – 2 Dec, 2001.
35. [FT07] Flow Tools Information. <http://www.splintered.net/sw/flow-tools/>. 2007.
36. [GRD03] Giorgio Giacinto, Fabio Roli, Luca Didaci. "Fusion of multiple classifiers for intrusion detection in computer networks". *Pattern recognition letters Volume 24, Issue 12*, pp. 1795-1803, August 2003.
37. [Hu04] Ken Hutchison. "Wireless Intrusion Detection System". *The SANS GSEC whitepaper published on GIAC Security Essentials, London, June 21-26, 2004*.
38. [HC03] Sang-Jun Han, Sung-Bae Cho. "Detecting Intrusion with rule based integration of multiple models". *Computers and Security, Volume 22, Issue 7*, pp. 613-623, October 2003.
39. [HHE+02] Andrew Honig, Andrew Howard, Eleazar Eskin, and Salvatore Stolfo. "Adaptive Model Generation: An Architecture for the Deployment of Data Mining-based Intrusion Detection Systems." *Data Mining for Security Applications. Kluwer 2002*
40. [HJI00] Hashim, S.J, Jumari, K, Ismail, M. "Computer network intrusion detection software development". *TENCON 2000. Proceedings , Volume: 3 , 24-27 Sept. 2000 Pages:117 - 123 vol.3*.
41. [HPJ03] Y. C. Hu, A. Perrig, and D. B. Johnson. "Packet leashes: a defense against wormhole attacks in wireless networks". *In proceedings of 22<sup>nd</sup> IEEE Annual Joint Conf. Computer and Communication Societies (Infocom 2003), Volume 3*, pp. 1976 – 1986, 2003.
42. [HPY+04] Jiawei Han, Jian Pei, Yiwen Yin and Runying Mao. "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach". *In journal of Data Mining and Knowledge Discovery Volume 8, pages 53-87, Number 1 / January, 2004*.

43. [ISA93] T. Imielinski, A. Swami, R. Agarwal. "Mining association rules between sets of items in large databases". *In proceeding of the ACM SIGMOD conference on management of data*, pp. 207 – 216, Washington D.C. May 1993.
44. [J05] Jensen, Kenneth G.; A combined association rule/radial-basis function neural network approach to intrusion detection; M.Sc Thesis, Dept. of Computer Science, Utah State University, 2005.
45. [JA00] Mahesh Joshi, Ramesh Agarwal. "A new framework for learning classifier model in data mining (a case study in network intrusion)". *In proceeding of the 1<sup>st</sup> SIAM Conference on Data Mining*, April 2000.
46. [JHK+01] Mahesh Joshi, Euihong Han, George Karypis, Vipin Kumar; Parallel Algorithms in Data Mining; Technical report TR 01-001, Department of Computer Science and Engineering, University of Minnesota, January 26, 2001. [http://www.cs.umn.edu/tech\\_reports\\_upload/tr2001/01-001.pdf](http://www.cs.umn.edu/tech_reports_upload/tr2001/01-001.pdf)
47. [KTK02] Christopher Krugel, Thomas Toth, Engin Kirda. "Service specific anomaly detection for network intrusion detection". *Proceedings of the 2002 ACM symposium on Applied computing*, pp. 201 – 208, March 2002.
48. [Linc07] Lincoln Laboratory, Massachusetts Institute of Technology. <http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/docs/attackDB.html>. 2007.
49. [Le02] Wenkee Lee. "Applying Data Mining to Intrusion Detection: the Quest for Automation, Efficiency, and Credibility". *ACM SIGKDD Exploration Newsletter*, vol. 4, issue 2, December 2002, pages 35-42.
50. [LB97] T. Lane, C.E. Brodley. "An application of machine learning to anomaly detection". *In proceeding of 20<sup>th</sup> National Information System Security Conference*, pp. 366 – 380, 1997.
51. [LFM+00] Wenke Lee, Wei Fan, Matthew Miller, Sal Stolfo, and Erez Zadok. "Toward Cost-Sensitive Modeling for Intrusion Detection and Response". *Workshop on Intrusion Detection and Prevention, 7th ACM Conference on Computer Security, Athens, GR*, pp. 5 – 22, November, 2000.
52. [LJ99] E. Lundin, E. Jonsson. "Some practical and fundamental problems with anomaly detection". *In: Proceedings of the Fourth Nordic Workshop on Secure IT Systems (NORDSEC'99), Kista, Sweden, 1-2 November 1999*.

53. [LJ00] Emilie Lundin, Erland Jonsson. "Anomaly-based Intrusion Detection: Privacy concerns and other problems". *Computer Networks, Volume 34, Issue 4*, pp. 623-640, October 2000.
54. [LLM+07] Y. Liu, Y. Li, H. Man, and W. Jiang. "A hybrid data mining anomaly detection technique in ad hoc networks". *International Journal of Wireless and Mobile Computing*, 2(1): pp. 37-46, 2007.
55. [LS00] Wenke Lee, Salvatore J. Stolfo. "A Framework for Constructing Features and Models for Intrusion Detection Systems". *ACM Transaction on Information and System Security*, vol. 3, no. 4, November 2000, pages 227-261.
56. [LSC+01a] Salvatore J. Stolfo, Wenke Lee, Philip K. Chan, Wei Fan and Eleazar Eskin. "Data Mining-based Intrusion Detectors: An Overview of the Columbia IDS Project". *ACM SIGMOD RECORD (4): Special Selection on Data Mining for Intrusion Detection and Threat Analysis*, Deniel Barbara, editor, pp. 5 - 14, Dec. 2001
57. [LSC+01b] Wenkee Lee, Salvatore J. Stolfo, Philip K. Chan, Eleazer Eskin, Wei Fan, Matthew Miller, Sholmo Hershkop and Junxin Zhang. "Real Time Data Mining-based Intrusion Detection". *Proceedings of DISCEX II, June 2001*
58. [LP99] U. Lindqvist, P. A. Porras. "Detecting Computer and Misuse Through the Production-based Expert System Toolset (P-Best)". *In proceedings of the 1999 IEEE symposium on security and privacy*. pp.146-161
59. [LS98] Lee, W., and S. Stolfo [1998]. "Data Mining Approaches for Intrusion Detection". *In proceedings of the 7<sup>th</sup> USENIX security symposium*, pp. 6 -6, San Antonio, TX
60. [LSF+00] W. Lee, S.J. Stolfo, W. Fan, A. Prodromidis, and P. Chan. "Cost sensitive modeling for fraud and intrusion detection: results from the JAM project". *In proceedings of the 2000 DARPA information survivability conference and exposition*, pp. 130 - 144, January 2000.
61. [LSM98] Wenke Lee, Sal Stolfo, and Kui Mok. "Mining Audit Data to Build Intrusion Detection Models". *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD '98)*, New York, NY, pp. 66 - 72, August 1998

62. [LSM99] W.Lee, S. J. Stolfo and K. Mok. "Data mining in workflow environments: Experiences in intrusion detection". *In proceedings of the 1999 conference on knowledge discovery and data mining (KDD -99), 1999*
63. [LXY03] Quing-Hua Li, Jia-Jun Xiong, Hua-Bing Yang. "An Efficient Algorithm for Frequent Pattern in Intrusion Detection". *Machine learning and cybernetics, 2003 International Conference on, Volume 1, 2-5 Nov. 2003. Pages: 138-142 vol.1*
64. [Me06] M. Metheny. "Wireless Intrusion Detection System (WIDS): Protecting the wireless perimeter". *A whitepaper published by Lunarline Inc. Washington DC, www.lunarline.com on 16<sup>th</sup> October, 2006.*
65. [Mob07] "What is ad-hoc network". [http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40\\_gci213462,00.html](http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci213462,00.html). 2007.
66. [MC03] V. Mahoney, Philip K. Chan. "Learning Rules for Anomaly Detection of Hostile Network Traffic". *Data Mining, 2003. ICDM 2003. Third IEEE International Conference, pp. 601, 2003*
67. [MCZ+00] Stefanos Manganaris, Marvin Christensen, Dan Zerkle, Keith Hermiz. "A Data Mining Analysis of RTID Alarms". *Computer Networks vol. 34, issue 4, October 2000, pages 571-577*
68. [MT04] H. Mannila and H. Toivonen. „Levelwise search and borders of theories in knowledge discovery“. *International Journal of Data Mining and Knowledge Discovery, 1(3):241–258, Jan 2004.*
69. [MT96] Manila, H., Toivonen, H. "Discovering Generalized Episode using minimal occurrences". *In proceedings of the 2<sup>nd</sup> International conference on knowledge discovery in databases and data mining, Portland, Oregon, pp. 146 – 151, August 1996.*
70. [MTV97] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo, "Discovery of Frequent Episodes in Event Sequences", *Data Mining and Knowledge Discovery 1(3): pp. 259-289 (1997)*
71. [NC03] Caleb C. Noble, Diane J. Cook. "Graph based Anomaly Detection". *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 631 – 636, August 2003.*
72. [NCD+02] Peng Ning, Yun Cui, Douglas S. Reeves. "Constructing attack scenarios through correlation of intrusion alerts". *Proceedings of the 9th ACM*

*conference on Computer and communications security, pp. 245 – 254, November 2002.*

73. [OB07] Open BSM: Open Source Basic Security Module (BSM) Audit Implementation. <http://www.trustedbsd.org/openbsm.html>. 2007.
74. [PES01] Leonid Portnoy, Eleazar Eskin and Salvatore J. Stolfo. "Intrusion detection with unlabeled data using clustering". *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*. Philadelphia, PA: November 5-8, 2001
75. [QW02] Yan Qiao, Xie Weixin. "A Network IDS with low false positive rate". *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. Volume 2, 12-17 May 2002. Pages 1121-1126.*
76. [Red08] Red-M – Superior Wireless Solution. <http://www.red-m.com/>. 2008.
77. [RB03] S. Radosavac and J. S. Baras, "Detection and Classification of Network Intrusions using Hidden Markov Models", 37th Conference on Information Sciences and Systems (CISS), Baltimore, March 2003.
78. [REA08] Ahmedur Rahman, C. I. Ezeife, A. K. Aggarwal. "WiFi Miner: An Online Apriori-Infrequent based Wireless Intrusion Detection System". In *2<sup>nd</sup> International Workshop on Knowledge Discovery from Sensor Data (Sensor-KDD 2008) in conjunction with the 14<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Las Vegas, USA, August 2008.*
79. [Sh04] Robert J. Shimonski. "Wireless Attacks Primer". *A whitepaper published on windowssecurity.com section: Articles: Wireless security in Jul 30, 2004.*
80. [Su96] A. Sundaram. "An Introduction to Intrusion Detection". *Crossroads: The ACM student magazine, 2(4), April, 1996.*
81. [SGF+02] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, S. Zhou. "Intrusion Detection: Specification based anomaly detection: a new approach". *Proceedings of the 9th ACM conference on Computer and communications security, pp. 265 – 274, November 2002.*
82. [SGV+99] R. Sekar, Y. Guang, S. Verma, T. Shanbhag. "A high performance network intrusion detection system". *Proceedings of the 6th ACM conference on Computer and communications security, pp. 8 – 17, November 1999.*
83. [SON95] A. Savasere, E. Omiecinski, and S. Navathe. "An Efficient Algorithm for Mining Association Rules in Large Databases". *In proceeding of 1995 Intl.*

*conference on Very Large Databases, pp. 432 – 444, Zurich, Switzerland, September 1995.*

84. [SZ02] Karlton Sequeira, Mohammed Zaki. “ADMIT: Anomaly based data mining for intrusions”. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 386 – 395, July 2002*
85. [Th05] Kurt Thearling. “An Introduction to Data Mining: Discovering Hidden Values in your Data Warehouse”. Whitepaper accessed in 2005 at <http://www.thearling.com/text/dmwhite/dmwhite.htm>
86. [TUA+98] Dick Tsur, Jeffrey D. Ullman, Serge Abiteboul, Chris Clifton, Rajeev Motwani, Svetlozar Nestorov and Arnon Rosenthal; “Query Flocks: A generalization of Association Rule Mining”; *In proceedings of the 1998 ACM SIGMOD Conference on Management of Data, pp. 1 – 11, Seattle, WA, June 2 – 4, 1998.*
87. [Wifi07] WiFi Planet. <http://www.wi-fiplanet.com/tutorials/article.php/1447501>. 2007.
88. [Wiki07] Wi-Fi Protected Access. [http://en.wikipedia.org/wiki/Wi-Fi\\_Protected\\_Access](http://en.wikipedia.org/wiki/Wi-Fi_Protected_Access). 2007.
89. [Wiki08] Wireless Intrusion Prevention System. [http://en.wikipedia.org/wiki/Wireless\\_intrusion\\_prevention\\_system](http://en.wikipedia.org/wiki/Wireless_intrusion_prevention_system). 2008.
90. [WZS02] H. Wang, D. Zhang, and K. G. Shin. “Detecting SYN flooding Attacks”. *In proceedings of 21<sup>nd</sup> IEEE Annual Joint Conf. Computer and Communication Societies (Infocom 2002), Volume 3, pp. 1530 – 1539, June 2002.*
91. [XQJ01] Yang Xiang-Rong, Song Qin-Bao, Shen Jun-Yi. “Implementation of sequence patterns mining in network intrusion detection system”. *Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences.*
92. [XSZ04] Hongxia Xia, Qi Shen, Luo Zhong, Shan Feng, Rui Hao. “Application of Data Mining Technology and generic algorithm to intrusion detection system”. *Proceedings of the 3rd international conference on Information security, pp. 218-219, November 2004.*

93. [Yo03] Kenichi Yoshida. "Entropy based Intrusion Detection". *Communications, Computers and Signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on, Volume: 2, 28-30 August 2003.*
94. [YGF+98] Nong Ye; Giordano, J.; Feldman, J.; Qiu Zhong. "Information fusion techniques for network intrusion detection". *Information Technology Conference, 1998. IEEE, 1-3 Sept 1998. Pages 117-120.*
95. [Zh02] Zhen Zhang; Data Mining Approach For Network Intrusion Detection; Presentation at the Department of Computer Science, California State University, Sacramento, 24<sup>th</sup> April, 2002. <http://gaia.ecs.csus.edu/~wang/ppt/ids.ppt>
96. [ZKN05] Zhong, S., Khoshgoftaar, T. and Nath, S. "A Clustering Approach to Wireless Network Intrusion Detection". *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05). pp.190-196.*
97. [ZS04] S Zanero, Sergio M Savaresi. "Unsupervised learning techniques for an Intrusion Detection System". *On proceeding of the 2004 ACM symposium on Applied computing, pp. 412 – 419, March 2004.*
98. [ZZW08] Hu Zhengbing, Li Zhitang, Wu Junqi. "A Novel Intrusion Detection System (NIDS) based on signature search of data mining". *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop, Adelaide, Australia, January 2008. Article no 45. Pages 237 – 243.*



## **Vita Auctoris**

<b>NAME</b>	S S Ahmedur Rahman
<b>PLACE OF BIRTH</b>	Dhaka, Bangladesh
<b>YEAR OF BIRTH</b>	1982
<b>EDUCATION</b>	Department of Computer Science University of Windsor, Windsor, Ontario, Canada B.Sc. (Honours) in Computer Information System (2002-2005)
	Department of Computer Science University of Windsor, Windsor, Ontario, Canada M.Sc. in Computer Science (2006-2008)