Wireless Intrusion Detection Sytem

John Louis Vigo Jr.

University of New Orleans

Follow this and additional works at: https://scholarworks.uno.edu/td

Recommended Citation
https://scholarworks.uno.edu/td/203

This Thesis is protected by copyright and/or related rights. It has been brought to you by ScholarWorks@UNO with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in University of New Orleans Theses and Dissertations by an authorized administrator of ScholarWorks@UNO. For more information, please contact scholarworks@uno.edu.
WIRELESS INTRUSION DETECTION SYSTEM

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
The Department of Computer Science

by

John Louis Vigo, Jr.

B.S., University of New Orleans, 1975
M.S., University of New Orleans, 1979

December 2004
# Table of Contents

List of Figures .................................................................................................................... iv  
Abstract ................................................................................................................................ v  
Chapter 1: Introduction ........................................................................................................ 1  
Chapter 2: IEEE Standard 802.11 ........................................................................................ 3  
  2.1 Overview .................................................................................................................... 3  
  2.2 Differences between wired and wireless LANs .......................................................... 3  
  2.3 802.11 architecture .................................................................................................... 4  
  2.4 802.11 services ............................................................................................................ 4  
    2.4.1 Authentication and privacy ................................................................................... 6  
  2.5 Relationships between services ................................................................................... 6  
  2.6 MAC Frame Format ................................................................................................... 7  
    2.6.1 Frame Control Format ........................................................................................... 8  
  2.7 IEEE 802.11i ............................................................................................................. 9  
  2.8 Other ........................................................................................................................... 9  
Chapter 3: Security problems in wireless networks ............................................................ 10  
  3.1 Ease of attack ............................................................................................................ 10  
  3.2 Objectives of security ............................................................................................... 10  
  3.3 Security services in IEEE 802.11 .............................................................................. 11  
    3.3.1 Weaknesses in authentication .............................................................................. 11  
    3.3.2 Weaknesses in WEP ............................................................................................ 11  
  3.4 Types of attacks ....................................................................................................... 12  
    3.4.1 Network discovery ............................................................................................... 12  
    3.4.2 Eavesdropping ..................................................................................................... 12  
    3.4.3 Man in the middle attack (MITM) ....................................................................... 13  
    3.4.4 Denial of service attacks (DOS) ......................................................................... 13  
    3.4.5 WEP key cracking ............................................................................................... 14  
    3.4.6 Other .................................................................................................................. 15  
  3.5 Need for Security ...................................................................................................... 15  
Chapter 4: Description of WIDS system ............................................................................ 16  
  4.1 General description ................................................................................................. 16  
  4.2 WIDS architecture .................................................................................................. 17  
  4.3 WIDS attack defenses ............................................................................................ 19  
    4.3.1 Network discovery and eavesdropping ............................................................... 19  
    4.3.2 Man in the middle attack (MITM) ....................................................................... 19  
    4.3.3 Denial of service attacks (DOS) ......................................................................... 19  
    4.3.4 WEP key cracking ............................................................................................... 19  
Chapter 5: Feasibility of WIDS ......................................................................................... 21
List of Figures

Figure 1: State variables and services................................................................................... 7
Figure 2: MAC frame format............................................................................................... 7
Figure 3: Frame control format........................................................................................... 8
Figure 4: WIDS architecture............................................................................................... 18
Figure 5: Signal Strength vs. Position.................................................................................. 31
Figure 6: Rules Tree............................................................................................................ 40
Figure 7: Change rules file............................................................................................... 45
Figure 8: Show Stats........................................................................................................... 46
Figure 9: WIDS log............................................................................................................. 47
Figure 10: Result of query “Agere” on WIDS log.............................................................. 48
Figure 11: WIDS Simulation Control GUI........................................................................... 49
Figure 12: WIDS simulation with sixteen simulated access points................................. 50
Figure 13: Reconfigured BETA AP and HOSTAP locating station................................. 55
Abstract

The decrease in price and the ease of use of wireless network devices make them an attractive alternative to standard wired networks. However, the intrinsic insecurity of wireless media and weaknesses in the standards for use of wireless media leave wireless networks vulnerable to attacks from unauthorized users. The intrinsic insecurity of wireless media results from radio signals extending beyond the networks intended coverage area and the weaknesses in the standards result from the methods used for authorization and privacy. These insecurities restrict the use of wireless networks by entities that need a high level of security. This paper describes a Wireless Intrusion Detection System (WIDS) that provides additional security for 802.11b wireless networks. WIDS provides intrusion detection that can react to potential threats and locate an intruder through the use of intelligent access points equipped with rotating directional antennas.
Chapter 1: Introduction

In November 2003, three Michigan men were indicted on federal charges in North Carolina for allegedly breaking into the Lowe’s home improvement chain’s central computer system and attempting to steal credit card numbers. According to the indictment, the men used a wireless computer card to access the wireless network of a Southfield, Michigan Lowe’s store and used that connection to connect to the Lowe’s central computer system in North Carolina and store systems at seven Lowe’s stores around the country. Software used to process credit card transactions was modified to save credit card numbers for later retrieval. Only six credit card numbers were collected. Two of the men told the FBI that they discovered the Southfield, Michigan Lowe’s wireless system while wardriving. The men were operating out of a 1995 Pontiac Grand Prix parked outside the Southfield store. It is not known if the Lowe’s wireless system was using encryption. [1], [2], [3]

Much has been written in recent years about the insecurity of wireless networks and 802.11. Attacks against wireless networks are not just theoretical. Software for attacking 802.11 networks can be downloaded from the Internet and there are numerous types of attacks ranging from simple network discovery to complex WEP key cracking. Attacks can be either active or passive. Searching for wireless networks by driving around in cars equipped with a laptop computer, a wireless card and a network discovery program has become so commonplace that it has been given the name “wardriving.”

Wireless networks are becoming more and more widespread. The cost of wireless access point/routers, wireless cards and other hardware for a wireless network continues to decrease and is readily available at electronic stores and over the Internet for less than $50 [4]. In September 2004, Comp USA advertised a Netgear 802.11b router and wireless USB adapter for a total price of $19.98 [5]. 802.11 wireless cards are included as standard networking devices on many laptop computers and PDA’s. As the price decreases home, business and governmental use of wireless networks is increasing.

However, the use of a wireless network comes with some built-in insecurities that are not present in wired networks. The wireless medium is broadcast by radio waves, which can be received by anyone within range who has the proper wireless hardware. The radio waves have no absolute boundaries and often extend beyond their area of intended coverage. No physical connection to the wireless network is needed. Those characteristics of a wireless network leave it open to active and passive attacks. Many of these networks use the 802.11b protocol.

While there are numerous wireless intrusion detection systems (IDS) available, most of them operate in the radio frequency (RF) monitor mode. An IDS operating in the monitor mode can see the frames in real time, but is not able to actively respond to identified threats, interact directly with an intruder or locate the intruder. Some save the monitored data and later look for anomalies
in the data to identify intruders [6]. Others just send alerts when a potential intruder is noticed [7]. If the data requires subsequent analysis to identify a potential problem, then the intruder could be gone before anyone knows he was there. An alert requires someone else or another program to take any action.

This paper presents a wireless intrusion detection system (WIDS) that provides additional security not found in other wireless intrusion detection systems. The WIDS provides security around the perimeter of a wireless network and identifies and locates potential intruders before they can connect to the internal network. The WIDS contains access points equipped with directional antennas that operate as intrusion detection devices, and the access points can interact with intruding stations, deny access, ignore frames, react to certain criteria, etc. The WIDS access points connect to a controller computer via a wired network. The controller computer configures the WIDS access points, shows statistics from stations associated with the WIDS access points, logs data sent by the WIDS access points, and locates intruder stations using data from two WIDS access points. The advantage of WIDS over other wireless intrusion detection systems is that WIDS can actively respond to potential threats and can locate the potential intruders before they can attack an interior network.

The following chapters present sections of the IEEE 802.11 standard “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications” that are most applicable to WIDS, security problems and attacks on IEEE 802.11b networks, a discussion of research done by others on locating 802.11 wireless stations, a more detailed description of WIDS, a partial implementation of WIDS, and test results of WIDS implementation. The paper ends with a conclusion and suggestions for future work.
Chapter 2: IEEE Standard 802.11

2.1 Overview

802.11 is part of a family of standards for local and metropolitan area networks dealing with the Physical and Data Link layers as defined by the International Organization for Standardization (ISO) Open Systems Interconnection (OSI) Basic Reference Model (ISO/IEC 7498-1: 1994). The 802.11 standards describe seven types of medium access technologies and related physical media.

Physical and data link standards for fixed, portable, and moving stations in a local area network (LAN) using the media of air, radio, or infrared are described in ANSI/IEEE Std 802.11, 1999 Edition entitled “Information technology -Telecommunications and information exchange between systems - Local and metropolitan area networks -Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.” [8]

The standard includes a protocol for authentication, association, and reassociation services, an optional encryption/decryption procedure, power management, a distributed coordination function, and a point coordination function for the bounded transfer of data.

Wireless devices that communicate by radio can use either a frequency-hopping spread spectrum (FHSS) supporting 1 Mbit/s and an optional 2 Mbit/s data rate or a direct sequence spread spectrum (DSSS) supporting both 1 and 2 Mbit/s data rates using the carrier sense multiple access protocol with collision avoidance (CSMA/CA) medium sharing mechanism. IEEE Std 802.11b-1999 modified IEEE Std 802.11 by extending DSSS to support 5.5 Mbit/s and 11 Mbit/s payload data rates.

2.2 Differences between wired and wireless LANs

There are substantial differences between wired and wireless LANs. IEEE802.11 explains some of these differences and describes ways to handle them. Some of the ways in which IEEE802.11 physical layers are different from wired media are:

- A destination address is not equal to a physical location. In wired LANs, an address represents a physical location. In 802.11, the addressable entity is a station. Stations are allowed to be mobile, so a station represents a destination that is not the same as a physical location.

- 802.11 media have no absolute or readily observable boundaries and are unprotected from outside signals.

- 802.11 must handle mobile and portable stations. Mobile stations access the LAN while in motion.
In order to appear as an IEEE 802 LAN to higher layers (logical link control), an 802.11 network must handle station mobility within the MAC layer.

The bottom two layers of the OSI model are the physical layer (layer 1) and the data link layer (layer 2). In 802.11, the data link layer has two sublayers - the Media Access Control (MAC) sublayer and the Logical Link Control (LLC) sublayer. The physical layer handles the physical layer convergence protocol (PLCP) and the physical medium.

2.3 802.11 architecture

The standard describes an 802.11 architecture for a wireless LAN that is made up of interacting components. The 802.11 architecture supports station mobility and is transparent to upper layers. The components of an 802.11 wireless LAN include:

- basic service set (BSS) - a set of stations controlled by a single coordination function. A BSS is the most basic component.

- basic service area (BSA) - the coverage area where the BSS members can communicate with each other.

- independent BSS (IBSS) - the most basic type of 802.11 LAN. It can consist of only two stations communicating with each other. An IBSS is often referred to as an ad hoc network.

- distribution system (DS) - connects multiple basic service sets to create an extended service set.

- extended service set (ESS) - a network of DS and BSSs.

- station (STA) - a device containing an 802.11 medium access control (MAC) and physical layer (PHY) interface to a wireless medium (WM). Stations include access points.

- access point (AP) - a station that provides access to the DS by providing DS services in addition to acting as a station. Data move between a BSS and the DS via an AP.

- portal - the logical point at which medium access control service data units (MSDUs) from an integrated non-IEEE 802.11 local area network (LAN) enter the distribution system (DS) of an extended service set (ESS). It is possible for a device to function as an access point and a portal.
2.4 802.11 services

802.11 does not specify how a DS is implemented. It specifies services associated with different components of the architecture. There are two kinds of 802.11 services – station service (SS) and distribution system service (DSS). Both are used by 802.11 MAC sublayer.

The station services are:

• Authentication - The authentication service is used by all stations to establish their identity to stations with which they will communicate. If authentication has not been established between two stations, association will not be allowed. A station may be authenticated with more than one station at a time.

• Deauthentication - The deauthentication service causes the station to be disassociated. Either authenticated party may send a deauthentication notice to the other party. The deauthentication notice cannot be refused by either party.

• Privacy - The privacy service provides confidentiality of data in the 802.11 LAN.

• MAC service data unit (MSDU) delivery - Information that is delivered as a unit between MAC service access points (SAPs).

The distribution system services are:

• Association - A station must associate with an access point before it is allowed to send a message via the access point. Association is always started by the mobile station and not by the access point. A station may be associated with only one access point at one time. An access point may be associated with numerous stations at one time.

• Disassociation - Either authenticated party may send a disassociation notice. The disassociation notice cannot be refused by either party.

• Distribution - The distribution service delivers medium access control service data units (MSDUs) within the distribution system. IEEE 802.11 does not specify how the message is distributed within the DS.

• Integration - The integration service enables delivery of medium access control service data units (MSDUs) between the distribution system (DS) and a non-IEEE 802.11 local area network. The details of an integration service depend on a specific DS implementation and are not covered by the standard.

• Reassociation - The reassociation service is invoked when a mobile station moves one AP to
2.4.1 Authentication and privacy

Access and confidentiality in 802.11 was designed to provide functionality equivalent to that which is inherent to wired LANs. Access and confidentiality in a wired LAN are controlled by the closed nature of the wired media. In a wired LAN, data passes through the wired media and is accessible only by devices physically connected to the network. In a wireless LAN, there is no physical connection to the LAN and any device within range with the correct type of hardware is able to receive data from the wireless LAN. Authentication is used in place of the physical connection to the wired LAN to provide access. Privacy provides the confidentiality of the wired LAN.

Authentication between stations is at the link level and is used to provide a wireless link equal to the assumed physical standards of a wired link. IEEE 802.11 uses authentication instead of a wired physical connection. Message origin to message destination, user to user, or per packet authentications are not provided by IEEE 802.11. Several authentication processes are supported including the open system authentication, which allows any station to become authenticated, and shared key authentication, which uses a shared, secret, wired equivalency privacy (WEP) key.

Privacy is provided by the wired equivalency privacy (WEP) algorithm. According to the standard, WEP “is designed to satisfy the goal of wired LAN ‘equivalent’ privacy. The algorithm is not designed for ultimate security but rather to be ‘at least as secure as a wire.”’ Privacy is provided for data frames and some authentication management frames. Initial data sent to set up the authentication and privacy services is sent in the clear. Unencrypted data frames received by a station expecting encrypted data frames are ignored.

2.5 Relationships between services

Each station keeps two state variables for each station it wants to communicate with. These are authentication state with values unauthenticated and authenticated, and association state with values unassociated and associated. The two state variables create three local states for each remote station. These are: state 1 - unauthenticated, unassociated; state 2 – authenticated, not associated; state 3 – authenticated and associated.
Figure 1: State variables and services

Each service is supported by one or more MAC message types. Three types of messages are used by the IEEE 802.11 sublayer - data, management and control. Data messages contain data. Management messages support the IEEE 802.11 services. Control messages support the delivery of the other two message types. Messages are sent via IEEE 802.11 frames.

2.6 MAC Frame Format

MAC frames contain a MAC header, a frame body, and a frame check sequence (FCS).

Figure 2: MAC frame format
The MAC header is made up of several fields:

- Frame control - details given later in section 2.6.1

- Duration/ID – For power save - poll frame subtype, this field contains the association identity (AID) of station that sent the frame. For all other frames, this field contains a duration value, which depends on the frame type.

- Addresses 1, 2, 3, and 4 – MAC addresses for sending and receiving messages. These fields are used for BSSID, destination, source, receiver, and transmitter addresses.

- Sequence control – Consists of a four bit fragment number and a twelve bit sequence number.

- Frame body – A variable length field containing information for the individual frame types and subtypes.

- FCS – contains a 32 bit cycle redundancy code (CRC).

### 2.6.1 Frame Control Format

An IEEE 802.11 frame control field has the following format.

![Frame control format](image)

**Figure 3: Frame control format**

- Protocol version – Identifies frames that are compatible with IEEE 802.11

- Type – Management, control, and data are allowed types

- Subtype – Together with type, identifies function of frame

- To DS – Set to ‘1’ if frame is going to the DS

- From DS – Set to ‘1’ if frame is coming from DS
• More Frag – Set to ‘1’ if there is another fragment

• Retry – Set to ‘1’ if frame is in retransmission

• Pwr Mgt – Set to ‘1’ if sending station is in power save mode

• More Data – Set to ‘1’ if more frames are to be transmitted

• WEP – Set to ‘1’ if WEP encryption is enabled

• Order – Set to ‘1’ if frames are strictly ordered

2.7 IEEE 802.11i

IEEE 802.11i, ratified in June 2004, is an addendum to IEEE 802.11. It was written to eliminate some of the problems with 802.11 such as WEP and authentication. It uses IEEE 802.1x, a port based access control mechanism, for user authentication and encryption key distribution. IEEE 802.11i improves encryption algorithms through the use of the temporal key integrity protocol (TKIP) and the counter mode with CBC=MAC protocol (CCMP). TKIP can be used with existing WLAN hardware and software upgrades. CCMP requires higher processing power and support for the advanced encryption standard (AES), which is available only in the latest WLAN hardware. [9]

802.1x restricts access to a network until the user has been authenticated by the network. 802.1x requires an authenticator, usually the AP in 802.11, a supplicant, the client in 802.11, and an authentication server that performs actual authentication of information provided by supplicant. The authentication server can be a separate device or reside in the authenticator.

IEEE 802.11i and 802.1x are discussed in detail in another paper on WIDS.

2.8 Other

IEEE Std 802.11 presents a lot of information. This paper presents some of the details of the IEEE 802.11 architecture, services, states, messages, and frames needed to design the WIDS intrusion detection rules and actions. It does not attempt to summarize the IEEE Std 802.11. For more details, the reader can find the original IEEE Standards at http://standards.ieee.org/getieee802/portfolio.html.
Chapter 3: Security problems in wireless networks

3.1 Ease of attack

Attacking a wireless network is easier than attacking a wired network. The radio signals of a wireless network often extend beyond the physical perimeter and can be picked up by anyone who has the correct hardware - a laptop computer and a wireless card are usually sufficient, and is close enough to the transmitter. A wireless network attacker does not have to be physically connected to the network through a wire. This eliminates an important security feature that is built into a wired network.

Wireless cards with the Intersil PRISM chipset can operate in a monitor mode. A station in the monitor mode can show all frames regardless of destination address, source address, FCS result, or WEP decryption result. This allows potential attackers to passively listen to traffic on the wireless network and obtain information such as MAC addresses, IP addresses, and BSSIDS, which can be used in various attacks on the network.

Some of the attacks rely on the weakness in IEEE802.11 that once a station is authenticated, no more authentication of 802.11 frames is done. An intruder can send disassociate or deauthenticate frames to disassociate or deauthenticate someone else. They can be sent to a broadcast address or to individual stations or access points.

Other attacks listen to 802.11 frames in transit and gather information, which can be used in other programs to spoof authorized users and gain access to wireless networks. 802.11 access points send out beacons that are not encrypted and 802.11 frames can be easily changed by software. For example, iwconfig can change essid, and setmac in airjack or ifconfig can change MAC addresses.

3.2 Objectives of security

Security should protect the integrity, confidentiality, non-repudiation, and availability of data. These objectives are defined as follows [10]:

- **Integrity – System**: performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system. Data: should be possible for receiver to verify that data has not been modified; intruder should not be able to substitute fake data.

- **Confidentiality – Only intended recipient(s) should be able to read data.**

- **Non-repudiation – Sender should not be able to falsely deny sending data.**
Availability – A third party with no access should not be able to block legitimate parties from using a resource.

802.11 has failed in meeting all of the above objectives. An attacker can easily include false data in an 802.11 frame and send it to an access point. An attacker can see wireless data if he has a wireless card that can operate in a monitor mode. Falsifying data in 802.11 and TCP/IP frames allows an attacker to send data that appears to be from someone else, and the attacker can later deny sending it. WEP has been found to be insecure. Legitimate parties can be denied access as a result of denial of service attacks performed at the physical layer by jamming the radio signals or at the MAC sublayer by sending a flood of deauthenticate or disassociate frames.

3.3 Security services in IEEE 802.11

Authentication services and WEP provide the security services in IEEE 802.11. The security is limited and not sufficient to stop an attacker from obtaining unauthorized access to wireless networks. The attacker then has the ability to read, intercept, change, or inject messages; to decipher encrypted messages; to disrupt or end authorized sessions; and to deny access to authorized users.

3.3.1 Weaknesses in authentication

Open system authentication provides little security since any station requesting Open System authentication may become authenticated if the recipient station is set to Open System authentication. Shared key authentication using WEP provides limited security due to weaknesses in WEP that allow the secret key to be passively recovered.

One approach to authentication, which is widely used but not required by IEEE 802.11, is MAC address filtering. The access point maintains a list of MAC addresses that are allowed to connect to the wireless LAN. During the connection process, the access point can check the station’s MAC address. If it is not on the access point’s list of MAC addresses, the station is not allowed to authenticate. The problem with Mac address filtering is that addresses identify stations, not users. It is very easy to monitor 802.11 frames in transit and recover valid MAC addresses. Software is available which can change a station’s MAC address.

3.3.2 Weaknesses in WEP

Weaknesses in WEP are well documented. A paper by Nikita Borisov, Ian Goldberg, and David Wagner "Intercepting Mobile Communications: The Insecurity of 802.11" identified some problems with WEP due to key stream reuse and weak message authentication [11]. Jesse R Walker’s paper "Unsafe at any key size; an analysis of the WEP encapsulation" showed that WEP is insecure regardless of the key size because of a weakness in WEP’s initialization vector [12]. A paper by Scott Fluhrer, Itsik Mantin, and Adi Shamir, "Weaknesses in the Key
Scheduling Algorithm of RC4," identifies weaknesses in the key scheduling algorithm and shows that rc4 is insecure in WEP. The authors describe but do not implement a passive cipher text only attack to recover a secret key [13]. Adam Stubblefield, John Ioannidis, and Aviel D. Rubin implemented the attack described by Fluhrer, Mantin, and Shamir and were able to recover a 128 bit secret key in a production network. Their implementation is described in their paper "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP [14]."

3.4 Types of attacks
There are several basic types of attacks – network discovery, eavesdropping, man in the middle attacks, denial of service attacks, and WEP key cracking. Attacks can be either active or passive.

3.4.1 Network discovery
The purpose of network discovery attacks is to find access points and obtain information about the network. Network discovery can do this actively or passively.

Software available for network discovery:

Netstumbler - actively discovers networks that broadcast their SSID by sending probe requests and receiving probe responses. Netstumbler displays the MAC address, SSID, name, channel, encryption and other information about access points. Global Positioning System (GPS) support is built in.

Wellenreiter - wireless card must be in monitor mode. It displays statistics, MAC address, WEP, and other data for ESSID of broadcasting and non-broadcasting networks. Wellenreiter requires the linux wlan-ng driver. GPS support is built in.

Kismet - wireless card must be in monitor mode. Kismet_server identifies name, BSSID (MAC address), WEP, channel and other information for networks.

3.4.2 Eavesdropping
Eavesdropping attacks listen to messages in transit and display the message data and/or save the message data to a file. Eavesdropping is also known as sniffing.

Software available for eavesdropping attacks:

Ethereal – wireless card must be in monitor mode for 802.11 frames. It shows a summary line, a protocol tree, and a hex dump. The protocol tree has a drill down function that provides details of the protocol fields.
Kismet – wireless card must be in monitor mode. It separates and identifies different wireless networks in the area. It saves data in a file format that can be read by ethereal.

HostAP - wireless card must be in monitor mode. HostAP has a sniffer called wlansniff, which displays 802.11 frame information.

3.4.3 Man in the middle attack (MITM)

In a man in the middle attack, stations connect to a rogue access point instead of the intended access point. The rogue access point intercepts data sent to/from the station from/to the intended access point. The rogue access point can read, change, or insert data.

Software available for man in the middle attacks:

monkey jack - Requires air jack driver. Forces target to deauthenticate from real access point and reassociate with attacking machine. The attacking machine associates with the real access point.

kracker-jack - Requires air jack driver. Improves monkey-jack to perform MITM attacks on IPsec connections.

Ettercap - an Address Resolution Protocol (ARP) attack. Ettercap can change the IP/MAC address combination in the target’s ARP cache to allow a MITM attack.

3.4.4 Denial of service attacks (DOS)

Denial of services attacks prevent users from accessing network resources. This is usually accomplished by flooding the network with faulty packets or messages that keep out legitimate traffic. Jamming of radio signals can also be used as a DOS attack.

Several papers on DOS attacks are worth mentioning because of the significance of the effects of attack or the methods used in performing the attack. The first two papers describe attacks for which there is no defense. The third paper describes an attack that alters fields overwritten by the card’s firmware.

In August 2004, researchers at Queensland University of Technology, Brisbane, Australia [15] discovered a weakness in the 802.11b standards that can be used to launch a DOS attack against wireless devices using 802.11b and DSSS. A test mode can be used to continually send a DSSS signal on a target channel to prevent the transmission of data from all access points and stations within range. The attack causes the wireless device’s Clear Channel Assessment (CCA) to report that the medium is always busy and consequently prohibiting the transmission of data. The attack targets the DSSS method used by 802.11b and 802.11g operating in mixed mode.
At the present time there is no defense against this type of attack other than not using 802.11b devices. The researchers state that 802.11b networking equipment can not be used to detect this attack because the monitor itself can be targeted. More expensive, specialized RF monitoring hardware may be needed for detection. The IEEE 802.11i standard will not prevent this attack since the attack operates at the Physical Layer Convergence Protocol (PLCP) layer and 802.11i protects MAC layer Protocol Data Units (PDUs).

A similar DOS attack used an $89.00 ATV transmitter operating in the same bandwidth as 802.11b and a 14dB gain directional grid antenna to send Radio Frequency Interference (RFI) to an access point [16]. The RFI causes the CCA to report that the current channel is busy just as the continuous DSSS signal did in the Australian attack.

A paper on DOS was presented at the 12th USENIX Security Symposium, August 4-8, 2003 [17]. The authors successfully performed DOS attacks by sending a spoofed deauthentication frame to an access point each time a data or association frame was received from a target station. The attack was carried out using an iPAQ H3600 Pocket PC and a Dlink DWL-650 PCMCIA wireless card. They also attempted a DOS attack that exploits vulnerability in the Network Allocation Vector (NAV), but it was unsuccessful due to the wireless cards improperly implementing the 802.11 MAC specifications. An interesting part of the NAV attack is that the authors figured out how to modify a frame after the firmware has processed it, but before it is sent. Thus, any field that is overwritten by the firmware can be altered before it is transmitted.

Altering fields overwritten by the firmware is significant because other intrusion detection methods analyze the sequence numbers, which are overwritten by the firmware, to identify frames generated by FakeAP and Airjack and MAC address spoofing [6]. If the sequence numbers can be changed, those detection methods will no longer work.

Software capable of performing DOS attacks

Wlan-jack - deauthentication request flood. Requires airjack driver

Void11 - association/authentication request flood

Fata-jack - sends invalid authentication requests to access point

FakeAP - flood area with beacon packets looking like hundreds of access points

3.4.5 WEP key cracking

WEP key cracking recovers the secret key used for encryption

Software capable of performing WEP key cracking attacks
AirSnort - Implementation of the Fluhrer, Mantin, and Shamir attack on WEP keys exploiting the weaknesses in the key scheduling algorithm of RC4. It requires 5 – 10 million encrypted packets [18]. It can break 40 bit or 128 bit keys. AirSnort requires a card that can operate in the monitor mode.

WEPcrack – Another implementation of the Fluhrer, Mantin, and Shamir attack on WEP keys.

3.4.6 Other

Impersonation or spoofing a valid station or access point in order to gain unauthorized access to the wireless network often uses information obtained from some of the above attacks, or it is part of an attack such as a MITM. Consequently, I did not list it as a separate type of attack.

3.5 Need for Security

As can be seen from the above discussion, an unprotected 802.11 network is vulnerable to many different kinds of attacks and it can sometimes be used as an entry point to a wired network as in the Lowe’s department store case mentioned in the introduction. Some kind of security or intrusion detection system should be used to protect sensitive data on a wireless network. WIDS can provide extra security to protect wireless 802.11b networks.
Chapter 4: Description of WIDS system

4.1 General description

The insecurities inherent in a wireless (no absolute or readily observable boundaries) coupled with the weaknesses in 802.11b security (authorization and WEP) make the use of 802.11b networks risky in places where a high level of security is needed.

WIDS is designed to strengthen the security of an internal wireless network by surrounding the physical perimeter of the network with special access points to detect and locate intruders. Radio signals from a wireless network extend out past the physical perimeter of the network and can be received by anyone with the correct equipment. Using information obtained about the wireless network from eavesdropping or by other means, a potential intruder could gain access to the wireless network and possibly a wired network behind the wireless network. There are situations when an entity may not want someone outside of the perimeter of the wireless network to have access to the network. Possible users of WIDS are military bases, ships in port, corporations with sensitive data, etc.

Authorized users connect to the internal network APs from within the perimeter. WIDS assumes that internal security procedures handle authentication inside the perimeter. Using the assumption that a station will associate with the access point from which it receives the strongest signal, a potential intruder outside the perimeter would associate with a WIDS access point (WIDS AP) on the perimeter and not with an access point in the internal network. The WIDS system would then locate the intruder using the directional antennas on two or more access points. Even if an intruder was able to associate with the internal access point from beyond the perimeter, the WIDS AP system directional antennas should still be able to locate the intruder, because WIDS can work with signals it picks up and association with a WIDS AP is not necessary.

WIDS is designed to give the user control over how it is applied. The user can decide that all stations outside the perimeter are intruders or that some stations outside the perimeter are authorized to connect to the WIDS APs or the internal network APs. Another possibility is for the WIDS APs to be used as honeypots that allow intruders to associate so that an intruder's activities can be monitored or the intruder can be given false or misleading data. The honeypot scenario is not implemented as part of this paper.

WIDS APs use a set of rules to identify potential intruders. The rules can search for known attack fingerprints or search according to user defined criteria based on the data in the signal (MAC address, IP header information, application data, etc.). The rules and the fact that WIDS contains access points give WIDS the ability to actively respond to identified threats. Each WIDS AP is equipped with a directional antenna which points outward and sweeps in a
preconfigured arc searching for wireless users. The WIDS AP determines if the remote station is
an intruder and it contacts a controlling computer. Other nearby WIDS APs are reconfigured by
the controller to modify their areas of coverage. Bearings from multiple WIDS APs are used by
the controller to pinpoint the location of the intruder. The rules can trigger actions to interact
with the intruder such as denying access, ignoring frames, reacting to IP header data, etc.

4.2 WIDS architecture

The WIDS architecture consists of one or more WIDS access points connected to a centralized
controller via a standard wired network. The wired network provides a more secure management
of the WIDS AP’s than a wireless network.

WIDS APs use an open-source access point code, HostAP and Linux. HostAP allows an 802.11b
wireless Prism 2/2.5 Ethernet card to be used as an access point instead of a remote station.
HostAP has been extended to include the additional functions needed by WIDS. The controller is
a personal computer running Linux. The controller software is written in Perl-TK.

We investigated several designs and configurations for the WIDS AP hardware and decided that it
should be an embedded computer equipped with a rotating directional antenna. The WIDS AP
would be built from components including an X-Scale single board computer and wireless
networking platform, a directional antenna, a stepper motor to rotate the antenna, and a stepper
motor controller. A list of the components is presented in the Appendix of this paper.
Unfortunately, we were not able to obtain financing to build a fully functional WIDS AP
equipped with the rotating antenna. A laptop computer without a rotating directional antenna
was used in our implementation.

Since some 802.11 frames that are handled by the Prism 2/2.5’s firmware when in hostAP can
only be captured by a wireless card in monitor mode, a configuration with two Prism 2/2.5 cards,
one in hostAP mode and one in monitor mode provides WIDS with more intrusion detection
capabilities. The card in the monitor mode can capture data, control, and the management frames
that are handled by the card’s firmware and pass them to the modified HostAP software for
processing and testing. The card in the monitor mode can also receive the hfa384x transaction
frames that show time, signal, silence and rate.
Figure 4 depicts four WIDS access points, arranged vertically on the left, protecting an internal wireless network installed in an arena. Only one side of a WIDS installation is shown in order to keep the picture simple. (A complete WIDS installation would include WIDS AP’s on all four sides of the arena.) The WIDS access points connect to a controller computer via a wired network. The controller can be located inside the perimeter. The arena contains six access points depicted as green circles emanating larger green circles. The larger green circles represent the signals transmitted by the AP’s. Blue arrows originating from each WIDS access point show the current orientation of an attached directional antenna. The other gray lines extending from two of the WIDS access points represent station signals picked up by the directional antennas. The red lines represent bearings on the station and a red square indicates the location of a station.

**Figure 4: WIDS architecture**

- WIDS access point
- Internal access point
- Station (possible intruder)
- Orientation of a WIDS AP attached directional antenna
- Estimates of an intruder’s location
4.3 WIDS attack defenses

If WIDS allows authorized users to associate with the WIDS APs or the WIDS APs are used as honeypots, then WIDS needs to protect the APs from the types of attacks mentioned in Chapter 3. This section discusses how WIDS identifies and handles some of those attacks.

4.3.1 Network discovery and eavesdropping

The WIDS AP antennas would not pick up network discovery and eavesdropping that are done passively. If network discovery or eavesdropping is done actively by sending probes or beacons, a WIDS AP with a card operating in the monitor mode could detect the 802.11 frames and identify the attack based on known fingerprints of the attacking software.

4.3.2 Man in the middle attack (MITM)

Man in the middle attacks need at least two devices in addition to the WIDS AP. Since one station masquerades as the other station, they will both send the same MAC address. The WIDS APs should pick up signals from both stations, even though only one station associates with the WIDS AP. If the two stations are picked up by different WIDS APs, the positions could be computed for each device in the normal manner. If the stations are in the same WIDS AP coverage area, the method used to compute the maximum signal strength position may need to distinguish the readings from each device using sequence control numbers or some other criteria.

4.3.3 Denial of service attacks (DOS)

If the DOS attack is a result of a flood of deauthenticate or disassociate frames, the HostAP software could keep track of how often they are received. A rule containing information such as “if x number of deauthenticate or disassociate frames are received in y time period, do not accept the deauthenticate or disassociate frame” could be implemented.

With regards to the attacks mentioned in section 3.4.4 that cause the CCA to report that the medium is always busy, it is not easy to determine how a WIDS AP would respond without reproducing the attack. The WIDS AP communicates with the WIDS controller via a wired medium so the attacks would not affect their communications. If the WIDS APs are able to interpret the signal strength of the jamming signal and report data similar to that of normal 802.11b DSSS frames, the controller should be able to locate the attacker. If the signal consists of noise that is not decipherable by the WIDS APs, then the WIDS would not be able to take a bearing on the attacker’s position.

4.3.4 WEP key cracking

The WIDS APs use open system authentication and do not normally use WEP. However, if
WEP is used and an attacker is passively trying to crack WEP, the WIDS AP antennas would not pick up the attacking equipment since no radio signals are sent out from the attacking device.
Chapter 5: Feasibility of WIDS

5.1 Locating intruders

Researchers at Interlink networks performed some experiments with 802.11 cards and access points and documented their results in a paper [19]. The purpose of their research and paper was to determine if it is possible to detect and locate unauthorized wireless clients and rogue access points with off the shelf components and free software. Using a Lucent Orinoco card in monitor mode, tcpdump and ethereal, the researchers were able to detect when a wireless card was powered on within range of a local network, a rogue access point and unauthorized clients. Using a Lucent Orinoco card in monitor mode, tcpdump and ethereal, the researchers were able to detect when a wireless card was powered on within range of a local network, a rogue access point and unauthorized clients.

The paper discussed several methods of locating an unauthorized client. One was to use a directional antenna and scan an area for the strongest signal, move the antenna and scan the area again for the strongest signal and use simple trigonometry to determine the location. Many factors can affect the signal such as terrain, obstructions, reflection, refraction, and atmospheric conditions. The result of these factors is that the apparent location of the strongest signal may not be the location of the signal. The researchers considered more discussion on this topic to be beyond the scope of their paper and no experiments were done using this method. However, the researchers believe the basic theory is sound and can be an effective technique to locate a transmitter.

Another method to locate a transmitter is to use the relative signal strength at several locations around the area, and if known, the propagation losses and the power of the transmitter. The first step is to compute distance vs. signal strength trends to predict the signal strength drop off rate and derive an equation to approximate indoor losses at 2.4 GHz. The next step is to write the distance-power approximation equation as a function of a position (x,y) and the unknown coordinates (A,B) of the transmitter and an unknown constant C where C is related to the transmitter’s output power.

Using a combination of experimental data and research on indoor radio propagation, the researchers derived the equations described above and tested their equations using three access points in an office suite with an area of approximately 30 by 20 meters. Three tests were presented in the paper. The results of two of the tests computed a location within a couple of meters of the actual position, but the third test computed a location that differed from the actual location by about eight meters. The researchers concluded that the basic theory is sound and that more careful data gathering and more data points would probably give better results. These results address locating an intruder within an indoor space.

Another paper by Frank Adelstein, Prasanth Alla, Rob Joyce and Golden G. Richard III [20] documents attempts to locate an intruding wireless station. In this paper the authors locate an intruder’s position by rotating a directional antenna 360 degrees while monitoring signal strength.
An antenna’s signature is computed ahead of time and the intruder’s location is computed by comparing the intruder’s data to the signature data.

Experiments were done with three different types of directional antennas. The test antenna was mounted on a telescope tripod and connected to a portable WIDS access point consisting of a laptop running Linux and HostAP. The antenna’s signature was computed by placing an intruder machine a few hundred feet away from the WIDS access point. The intruder associated with the WIDS access point. The directional antenna was rotated in fixed increments and the signal strength value was recorded. Readings were taken over a 360 degree range.

In order to compute the location of an intruder, several readings of the intruder’s signal strength data were made at different angles of the WIDS access point antenna. The intruder’s data was correlated with the signature data and the computed angle is the angle with the best fit to the known signature data. Two different locations of the WIDS access point were used. The results of the tests computed the location of the intruder within 2 to 4 degrees. The intruder’s location was computed using triangulation with the computed angles and access point locations.

The research showed that it is possible to determine the intruder’s location with reasonable accuracy using low priced off the shelf hardware and free software.

5.1.1 Comparison of WIDS and above experiments

The WIDS system is designed to operate with free software and off the shelf or slightly modified equipment. The above research shows that it is possible to locate 802.11 stations using free software and off the shelf or slightly modified equipment.

While the research in the first paper showed that it is possible to detect unauthorized clients and rogue access points, it has several deficiencies that need to be addressed by an intrusion detection system such as WIDS. One is that detection is not done in real time. The packets are captured to a file and analyzed later with ethereal. Another is that the wireless card used for detection is in monitor mode and it doesn’t communicate with the intruder or access point. Those deficiencies make it difficult to take action against an unauthorized user on a timely basis. The unauthorized user could connect to the wireless network, do whatever he wants and disconnect before anyone knows he was there. Also, the intruder was located in an interior space and WIDS is concerned with locating intruders in external locations. The second paper locates intruders in external locations but requires plenty of data collection for each access point.

WIDS is designed to use rotating directional antennas that scan an area for the strongest signal. With data from two access points with directional antennas, trigonometry can be used to determine the location. This is the method discussed in the first paper but not implemented. Since no directional antennas were available due to lack of funds, it was not possible to test the WIDS method for locating intruders.
5.2 Intersil Prism 2/2.5 802.11b wireless cards and HostAP driver

The WIDS access points require a wireless card using the Intersil PRISM 2/2.5 radio chipset. The PRISM Medium Access Controller (PRISM MAC) has two major operating modes for supporting access points: firmware based or host-based. The advantage of using a host-based access point is that the developer of the host access point software is able to implement optional features. The WIDS access point software needs additional features to control the sweeping directional antenna, apply a set of intrusion detection rules, and communicate with the WIDS controller. Our approach to implementing the WIDS access point software modifies the host-based access point driver HostAP.

5.2.1 PRISM 2/2.5 Host-based Access Point Mode

In the PRISM 2/2.5 host-based access point mode, the host software performs many functions required to maintain a Basic Service Set (BSS). These functions include authentication, deauthentication, association, reassociation, disassociation, data transmission between two wireless stations, and all power save queuing. When in host access point mode (hostAP), station firmware performs only packet transfer and low level 802.11 MAC management. Frame transmission, frame reception and beacon/probe response transmission are implemented in the firmware due to their time critical nature. The remainder of the access point operations is performed by the host software. The firmware acts as a frame pipe that transmits 802.11 frames from the host and receives 802.11 frames and passes them to the host. The hostAP mode is enabled by setting the PortType for station firmware to hostAP.

For transmitted frames, the host specifies the Protocol Version, Type, Subtype, To DS, From Ds, Pwr Mgt, More Data, WEP, and Order subfields and the firmware provides the More Frag and Retry subfields of the frame control field. The firmware fills in the Frame Control, Address 2, and Sequence Control fields of the MAC frame.

5.2.2 The PRISM MAC communications frame structure

The PRISM MAC communications frame structure consists of control fields, header information, and data segments. The control fields segment contains frame processing and status information and it is handled by the PRISM MAC controller. The header information and the data segments are transferred to the host. The control fields for a Transmit Frame Structure are the Status, SwSupport, TxRate, RetryCount, TxControl fields. The control fields for a Receive Frame Structure are the Status, Timestamp, Signal, Silence, and Rate fields.

The PRISM MAC frame structure handles both 802.3 and 802.11 header formats. A transmit frame structure can contain the header info in either 802.3 or 802.11 format. If the header is an 802.3 header, the PRISM MAC expands it into an 802.11 header. A receive frame structure contains the header info in 802.11 and 802.3 formats.
If higher software layers can handle 802.11 frame formats, the host software uses the 802.11 format part of the header info. If higher software layers assume an 802.3 frame format, the PRISM MAC handles the differences between the 802.11 and the 802.3 frame formats without involving the host software.

### 5.2.3 Monitor Mode

The PRISM 2/2.5 cards can operate in a non-standard monitor mode. The monitor mode is intended to be used for testing and debugging operations. However, it can also be used to monitor wireless networks. If a station is operating in monitor mode, all frames where the physical layer convergence protocol (PLCP) header and the beginning of the frame are successfully received are passed to the host regardless of destination address, source address, FCS result, or WEP decryption result.

### 5.2.4 HostAP software

HostAP [21] is a Linux driver for wireless LAN cards based on Intersil's PRISM 2/2.5 chipset. It implements the host-based access point functions discussed above. In addition, the HostAP driver has various features for development and debugging in IEEE 802.11 environments, including access to hardware configuration records, I/O registers, and frames with 802.11 headers.

HostAP includes a utility sniff, for monitoring activities. Sniff shows the data from the PRISM2/2.5 RX frames and the 802.11 frames. HostAP can operate in a kernel module or a user space daemon. WIDS uses the user space daemon.
Chapter 6: Software implementation of WIDS

6.1 WIDS Access Point/Controller Protocol

The WIDS access points and the controller communicate over TCP, using the WIDS Access Point/Controller Protocol. This section presents a summary of the communications between access points and the controller. Access Point/Controller Protocol message types are printed in upper case letters.

An access point starts up and attempts to connect to the controller using sockets. If the access point is unable to connect, it retries every 5 seconds. When a connection is made, the access point sends a HELLO message to the controller and waits for a response. The HELLO message contains the name of the access point and its location. The controller replies with a CONFIG message containing the beginning and ending sweep positions for the antenna, the report rate for sending data to the controller and the intrusion detection rules to be used by the access point. The antenna begins sweeping and listening for signals. The report rate specifies how often the access point sends REPORT messages to the controller. The REPORT message contains the access point name, the position (angle) of the antenna, the sweep direction, the time, the signal strength received at that position, and the MAC address of the station sending the signal.

When a signal is received which has the maximum signal strength in a specified number of readings, the access point sends a POSITION message to the controller. The POSITION message contains the access point name, the position of the antenna when the maximum signal strength was received, the MAC address of the station sending the signal and whether or not the station is associated with the reporting access point.

The controller maintains a hash of POSITIONS received. The keys are the MAC addresses of the stations and the values are strings containing the name of the access point that sent the POSITION message, an integer indicating whether or not the access point and the station are associated, and another integer indicating whether or not the station has been located.

When a POSITION message is received by the controller, the controller checks the hash for the MAC address. If it is not found, the controller adds it to the hash, and checks if the station is associated with the reporting access point. If it is associated, the controller determines which neighboring access point needs to be reconfigured and sends a RECONFIG message with reconfigured antenna sweep information to the applicable access point.

If the MAC address is in the hash, the controller checks the hash value to determine if the station has already been located or if a POSITION message from the same access point with the same information has been previously received. If so, the controller does nothing. If the MAC is in the hash with a different access point and has not been located, the controller will use the information
from both access points to compute the station’s location. The hash is updated to show that this station has been located.

The controller will display a warning message if the same MAC address is associated with more than one access point. This may be an indication that MAC address spoofing has occurred.

When an access point receives a RECONFIG message, it changes the starting sweep position or the ending sweep position in accordance with the RECONFIG message. The antenna sweeps to the new position twice and then returns to its original sweep configuration.

The controller operator can send messages to an access point to change the intrusion detection rules or to collect statistics related to a station associated with the access point. A CHANGERULES message is sent to change the intrusion detection rules. When the access point receives a CHANGERULES message, it destroys its current rule tree and processes the new rules sent by the controller. A STATS message is sent to gather statistics from an access point. The access point returns a STATS message containing a list of stations associated with it in response to the controller’s STATS message. The controller operator selects a station and sends a MAC message with the station’s MAC address. The access point reads the applicable /proc/net/hostap/wlan0/MAC address file and sends the stats info back to the controller in a MACST message. The controller receives the MACST message, opens a new window and displays the stats.

The access point can send a LOGDATA message to the controller. When the controller receives a LOGDATA message, it writes the message to the controller’s log file.

6.2 Intrusion Detection Rules

Each 802.11b frame received by a HostAP access point which is not handled by the wireless card’s firmware can be tested with a set of intrusion detection rules. If the access point also has a wireless card operating in the monitor mode, all 802.11b frame received can be tested with the rules. The results of the rules tests are intended to identify potential intruders.

A rule has the format type-subtype, a test list consisting of one or more pairs of test and test data, and one pair of action and action data. The type-subtype is the 802.11 frame type-subtype, the test is one of the tests defined in the WIDS AP software, the test data is data used in the test, the action is one is one of the actions defined in the WIDS AP software, and the action data gives more information about the action. The action is performed only if all of the tests in the rule are true.

The type-subtype, test list, and action are separated by ‘,’. The test list is enclosed by ‘(‘ and ‘)’. The test keyword and test data are separated by a ‘:’. The end of a test keyword test data
pair is designated by an ‘!’ . An action consists of an action keyword and action data pair. The action keyword is separated from the action data by a ‘:’.

The test keyword generally describes an operation on a field in the 802.11 or the IP frame, such as SOURCE_MAC_EQUALS for source MAC address equals, TTL_LESS_THAN for time to live is less than, or FLAG_PWRMGMT for power management is on or off. The value to be tested is given by the rule’s test data. Sample test data for test keywords are 00022D2E5C6a for SOURCE_MAC_EQUALS, 64 for TTL_LESS_THAN and OFF for FLAG_PWRMGMT.

An example of a rule is:

"MGMT_ASREQ, (SOURCE_MAC_EQUALS:00022D2E5C6a! FLAG_WEP:OFF!), LOG, assoc request from 00:02:2D:2E:5D:6a  flag wep off"

This rule is testing an association request. It is true if the source MAC address is 00:02:2D:2E:5C:6a and the wep flag is 0. If the rule is true, the WIDS AP will send a message to the controller, which displays the message in a terminal window and logs the message in a file. The message includes the WIDS AP name, date and time of message, and the action data "assoc request from 00:02:2D:2E:5D:6a  flag wep off."

6.3 Changes/additions to HostAP files

6.3.1 Changes to HostAP files

The first step in implementing WIDS in HostAP is to modify the struct hostapd_data in hostapd.h to include information for the directional antenna, the location of the access point, and the shared memory. HostAP defines the typedef hostapd for the struct hostapd_data. Information added to hostapd_data is:

```c
int controller_sock;    /* socket for controller*/
int beginsweep;        /* begin sweep position of directional antenna */
int endsweep;          /* end sweep position of directional antenna */
int position;          /* sweep position of directional antenna */
int x;                 /* x coordinate of access point location */
int y;                 /* y coordinate of access point location */
int reportrate;        /* frequency of reporting to controller */
int direction;         /* direction of antenna sweep-clockwise, counterclockwise */
int semid;             /* semaphore id */
int shmid;             /* shared memory id */
char *shm_ptr;         /* pointer to shared memory */
struct sharedinfo *shared_ptr; /* pointer to shared memory data struct*/
```

27
The sharedinfo structure holds the data shared between the hostapd process and the IP packet monitoring process. It is:

```c
struct sharedinfo{
    u8 ttl;             /* time to live */
    u8 src_mac[6];      /* source MAC address*/
    u8 dest_mac[6];     /* destination MAC address*/
    u16 seq_number;     /* sequence number and fragment number*/
    u32 test;           /* rule test to be performed */
    int testmac;        /* MAC address tested */
    pid_t pid;           /* parent process id */
    struct rule_node *ruledetail; /* rule */
    char message[MESSAGE_SIZE];   /* additional info, if needed */
};
```

The struct sta_info in ap.h also has to be modified to include information about the directional antenna. Additions are:

```c
int count;                /* number of completed sweeps */
int x;                    /* x coordinate of access point location */
int y;                    /* y coordinate of access point location */
int angle;                /* sweep position of directional antenna */
int signal;               /* signal strength */
int widsassociated;       /* associated with wids controller */
time_t *gmt;              /* time */
```

The hostapd configuration file must be changed to include information needed by the access point at start up. This information includes names of files to be read, name and location of log file, name of access point, location of the access point, and IP addresses and ports used by the access point and the controller. A sample of the information added to the configuration file is given below:

```plaintext
valid_mac_file=oui.txt1 /* file containing valid OUI numbers for MAC addresses */
rules_hash_file=80211a.txt /* file containing key(integer) value(text) pairs for rules */
log=log.new /* name of log file for access point logging */
log_directory=/home/jvigo/hostap-0.0.3/hostapd2new /* directory of log file */
ipaddress=127.0.0.1 /* access point IP address */
port=7000 /* access point port */
apname=HOSTAP /* access point name */
cipaddress=192.168.1.2 /* controller IP address */
cport=7777 /* controller port */
```
Changes to programs config.h and config.c are needed to handle the above additions to the configuration file. In config.h the struct hostapd_config is modified to include character pointers for each of the above items and config.c is expanded to include code to read the new items included in the configuration file.

The controller needs to know all of the stations associated with an access point, so a function char * ap_list_sta_str(hostapd *hapd) is added to sta_info.c. This function returns a string of all the associated stations for a particular access point.

In order to keep modifications to HostAP files at a minimum, functionality was put into separate files and calls to the functions were inserted into the HostAP files where necessary. The function calls inserted into the main function in hostapd.c initialize the shared memory, connect to the controller, open a log file, and load the hash tables for the MAC addresses and the rule descriptions. Some changes were made to the signal handling functions for SIGTERM and SIGINT, and a new function was written which is called when a SIGUSR2 signal is received. The file eloop.c, which contains the run loop called by main in hostapd.c, was modified to include a call to antenna_ap() in client.c. antenna_ap() handles the directional antenna.

The next step was to modify HostAP files to apply the intrusion detection rules. Management type/subtype rules are handled by adding a call to applyRules(struct ieee80211_mgmt *mgmt, hostapd *hapd) in each of the following functions in ieee802_11.c.

static void handle_auth(…)
static void handle_assoc(...)
static void handle_assoc.resp(...)
static void handle_disassoc(...)
static void handle_deauth(...)
static void handle_beacon(…)

Control and data type/subtype rules are handled by adding calls to applyRules(…) to receive.c functions static void handle_frame(hostapd *hapd, char *buf, size_t len) and static void handle_data(hostapd *hapd, char *buf, size_t len, u16 stype).

A rule returns true if all of the rules criteria specified in the tests are met and false if one or more of the criteria are not met. When a rule tests IP packets and a true result is obtained, a SIGUSR2 signal is sent by the process monitoring the IP packets to the HostAP process. The function packetresult() is called when a SIGUSR2 signal is received.
6.3.2 New files added to HostAP

Several new files were written to handle operations not included in Hostap. These files are antenna.c, client.c, ruleslist.c, validmac.c and their related header files.

6.3.2.1 antenna.c

The file antenna.c simulates antenna readings. Since there is no rotating directional antenna connected to the access point, antenna readings are read from an array where the array index represents the antenna position in degrees. Antenna.c tests the signal strength readings and finds the maximum signal strength within a range of readings reported by the sweeping directional antenna.

Some of the more important functions in antenna.c are:

6.3.2.1.1 void maxsignal (int position, int signal, struct pos_sig *max2){ … }

Maxsignal finds the maximum signal strength in a range of signal readings. The number of readings needed by the antenna before a maximum signal strength can be determined is designated by the variable POINTS. The maximum signal strength is calculated when there are POINTS/2 signal readings with an increasing trend on one side of a location and POINTS/2 signal readings with a decreasing trend on the other side of the same location. Temporary fluctuations in a range of increasing or decreasing signal strengths do not change the overall trend. For example, if signal strengths readings are 40, 42, 43, 41, 43, 44, the number of increasing signal readings is 6, since 41 is considered a temporary fluctuation and the trend is still increasing.

More than one station in a WIDS AP’s coverage area can be simulated as long as the sets of POINTS used to compute each local maximum do not overlap. The algorithm to compute the maximum signal strength position assumes all signal readings in the range of POINTS come from the same station. In a real WIDS implementation, the algorithm could try to distinguish signals from different stations using MAC addresses or sequence numbers, although either of those could be spoofed by a knowledgeable attacker.

The maximum is stored in the struct pos_sig.

```c
struct pos_sig{
    double slope; /* angle position where slope of curve = 0*/
    double signal; /* signal strength */
};
```

In order to compute the location of the maximum signal strength, the equation of a parabola is determined using the least squares method and the signal strength readings in the range of...
The maximum signal strength location is the maximum point on the parabola that is the point where the slope of the parabola is equal to zero. The maximum signal strength calculation is performed by calling the function double slope().

Slope() determines the matrices needed to solve the set of equations needed to fit a least squares parabola through a set of points which includes a maximum signal strength. Slope() calls the function solve(a_data, b_data, parabola) to get the equation of the parabola. Slope() then calculates the maximum point of the parabola by solving the equation where the derivative of the parabola equation equals zero and returns the maximum point. The derivative f ’(a) = Slope of tangent line to the graph of f at the point (a, f(a)).

Applying the method of least squares to curvilinear relationships gives a regression equation of the form

\[ \bar{Y}_X = aX^2 + bX + c \]

which is the equation of a parabola. \( \bar{Y}_X \) represents the values obtained from the estimated
regression equation. The values of a,b,c may be determined from a solution of the equations [22]:

\[\begin{align*}
\sum X^2 Y &= a \sum X^4 + b \sum X^3 + c \sum X^2 \\
\sum XY &= a \sum X^3 + b \sum X^2 + c \sum X \\
\sum Y &= a \sum X^2 + b \sum X + nc
\end{align*}\]

The equations can be written in matrix format as:

\[
\begin{bmatrix}
\sum X^4 & \sum X^3 & \sum X^2 \\
\sum X^3 & \sum X^2 & \sum X \\
\sum X^2 & \sum X & n
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
\sum X^2 Y \\
\sum XY \\
\sum Y
\end{bmatrix}
\]

\begin{verbatim}
slope() stores equations in arrays as:

    a_data[0] = sumx4; /* \sum X^4 */
    a_data[1] = sumx3; /* \sum X^3 */
    a_data[2] = sumx2; /* \sum X^2 */
    a_data[3] = sumx3; /* \sum X^3 */
    a_data[4] = sumx2; /* \sum X^2 */
    a_data[5] = sumx; /* \sum X */
    a_data[6] = sumx2; /* \sum X^2 */
    a_data[7] = sumx; /* \sum X */
    a_data[8] = POINTS; /* number of points */

    b_data[0] = sumx2y; /* \sum X^2 Y */
    b_data[1] = sumxy; /* \sum XY */
    b_data[2] = sumy; /* \sum Y */

parabola[0] = a,
parabola[1] = b,
parabola[2] = c,

and then calls
    solve (double a_data[], double b_data[], double parabola[]) to compute a, b, and c.
\end{verbatim}
The derivative of the equation for the parabola \( ax^2 + bx + c \) is \( 2ax + b \) and the maximum point in the parabola is the point \( x \) where the derivative = 0.

Slope() then solves the equation \( 2.0*\text{parabola}[0]*x + \text{parabola}[1] = 0 \) for \( x \):

\[
x = \frac{-\text{parabola}[1]}{2.0 * \text{parabola}[0]};
\]

6.3.2.1.2 void solve (double a_data[], double b_data[], double parabola[]){…}

The GNU Scientific Library (GSL) Linear Algebra library is used to solve the set of equations for the least squares parabola solution discussed above. It solves the linear system \( \mathbf{A} \mathbf{x} = \mathbf{b} \) using LU decomposition of the matrix \( \mathbf{A} \).

The code for solve(double a_data[], double b_data[], double parabola[]) is taken from File: gsl-ref.info, Node: Linear Algebra Examples, in the GSL documentation.

```c
void solve (double a_data[], double b_data[], double parabola[]){
    /* The solution is stored in  gsl_vector *x */
    int s;

    gsl_matrix_view m = gsl_matrix_view_array (a_data, 3, 3);
    gsl_vector_view b = gsl_vector_view_array (b_data, 3);
    gsl_vector *x = gsl_vector_alloc (3);
    gsl_permutation *p = gsl_permutation_alloc (3);
    gsl_linalg_LU_decomp (&m.matrix, p, &s);
    gsl_linalg_LU_solve (&m.matrix, p, &b.vector, x);
    gsl_permutation_free (p);

    parabola[0] = x->data[0];
    parabola[1] = x->data[1];
    parabola[2] = x->data[2];
}
```

Since there is no rotating directional antenna connected to the access point, the antenna readings must be simulated. When a station associates with an access point, the simulated antenna readings for that access point are read from a file into an array. The array index is the angle position in degrees. Valid angle values are –360 to 360. If an angle value is less than zero, a conversion factor is applied to it so that the array index is greater than or equal to zero.
6.3.2.1.3 int fgetangles (char *angles[], int n1, char *name){…}

The function fgetangles reads simulated antenna data from the file ‘name’. The format for an antenna reading is degrees, signal strength, MAC address. The first number represents the angle in degrees. The second number represents the signal strength at that angle and the next 6 numbers represent the MAC address of the station transmitting the signal. Every two digits of the MAC address are separated by a ‘:’. Each reading in the file must be separated by ',' or '
' and there must be a '
' before the EOF character. Comments start with '#' and the '#' is allowed only at the start of a line. The following is a valid file for antenna readings.

File1:

-35,10;00:02:2D:2E:5C:61,-34,12;00:02:2D:2E:5C:61,-33,16;00:02:2D:2E:5C:61,-32,20;00:02:2D:2E:5C:61,-31,23;00:02:2D:2E:5C:61,-30,25;00:02:2D:2E:5C:61,-29,27;00:02:2D:2E:5C:61,-28,24;00:02:2D:2E:5C:61,-27,27;00:02:2D:2E:5C:61,-26,28;00:02:2D:2E:5C:61,-25,29;00:02:2D:2E:5C:61,-24,26;00:02:2D:2E:5C:61,-23,27;00:02:2D:2E:5C:61,-22,25;00:02:2D:2E:5C:61,-21,23;00:02:2D:2E:5C:61,-20,20;00:02:2D:2E:5C:61,-19,16;00:02:2D:2E:5C:61,-18,14;00:02:2D:2E:5C:61,-17,12;00:02:2D:2E:5C:61

The first antenna reading is angle = -35, signal strength is 10 and the MAC address of the station is 00:02:2D:2E:5C:61.

Simulation of an access point receiving signals from more than one station can be done by using multiple MAC addresses in the antenna data file.

In a non-simulated environment with directional antennas HostAP can get signal strengths from the applicable /proc/net/hostap/wlan0/MAC address file or by using the Linux command iwspy from the wireless tools package [23]. If the WIDS AP has a wireless card in the monitor mode, HostAP can get the signal strength from the rx header.

6.3.2.2 client.c

Client.c handles connecting to and communicating with the controller. It sends and receives messages to/from controller, reports the antenna’s position to the controller, and reports the antenna’s position with the maximum signal strength of a station to the controller. Client.c also does the simulated positioning of the directional antenna.

Some of the more important functions in client.c are:
6.3.2.2.1 int connectToController(hostapd *hapd){ …}

ConnectToController uses sockets to connect to the controller. The access point name and the controller IP address and port are given in the hostapd.conf file and stored in the typedef struct hostapd. When there is a successful connection with the controller, client.c calls the eloop.c function eloop_register_read_sock(int sock, void (*handler)(int sock, void *eloop_ctx, void *sock_ctx), void *eloop_data, void *user_data){…} with the socket file descriptor, handleControllerMessage, hapd, and NULL as arguments to register handleControllerMessage as a handler for read events.

6.3.2.2.2 int associate(hostapd *hapd, struct sta_info *sta){…}

When a station associates with a HostAP access point, associate(…) calls the function getantennareadings(hapd->beginsweep) to read the simulated antenna data and writes a message to the access point log which lists the access point name, the date and time, the MAC address of the station and the station’s wireless card type. A sample message is:

    HOSTAP Thu Aug 26 02:48:20 2004 associated with 00:0d:88:4b:e0:d6 wireless card
    D:Link Corporation

The HostAP access point sends the same message to the controller.

6.3.2.2.3 void notifyController(int sock,char buffer1[], hostapd *hapd) {...}

NotifyController sends HELLO message to controller and waits for response.

6.3.2.2.4 static void handleControllerMessage(int sock, void *eloop_ctx, void *sock_ctx) {...}

When a message is received from the controller, HostAP’s run loop function calls handleControllerMessage with the socket file descriptor and two void pointers *eloop_ctx and *sock_ctx as arguments. HandleControllerMessage(…) casts *eloop_ctx as a *hostapd, initializes a buffer, and calls int recmesg(int fd, char buffer1[], hostapd *hapd) to read and process the message.

6.3.2.2.5 void antenna_ap(struct hapd_interfaces *ha){…}

Antenna_ap(…) handles the simulated positioning of the directional antenna and reporting the position to the controller. Each pass through the HostAP run loop calls antenna_ap(…) to move the directional antenna to its next position and reports the new position to controller, change sweep direction if necessary, return reconfigured sweep positions to normal if necessary, and call testantenna(…) in antenna.c to check if a maximum signal strength reading has been found. If so,
it calls the function void sendapfound(hostap *hapd, struct pos_sig *sig, char macStr[]) to
notify the controller of the antenna’s position with the maximum signal strength reading and the
MAC address of the station sending the signal.

6.3.2.2.6 int sendmesg(const char *arg,...){…}

Sendmesg(...) sends a message to controller. Sendmesg() takes a variable argument list of const
char *arg. The last word of arguments must be ’done.’

6.3.2.2.7 int recmesg(int fd, char buffer1[], hostapd *hapd){…}

Recmesg(...) reads and processes a message received from controller.

6.3.2.3 ruleslist.h

The ruleslist.h and ruleslist.c files are the longest and most important files added to HostAP.
Ruleslist.c processes the intrusion detection rules received from the controller and stores them in
a tree data structure. When an 802.11 frame is received, ruleslist.c searches the tree for rules with
matching type-subtype. If a rule is found, ruleslist.c performs the tests contained in the rule. If
the results of the tests are true, ruleslist.c performs the action in the rule. Ruleslist.h defines the
rule tests and actions key words and the structs used to hold the rules and the rule tree nodes.
Ruleslist.h also contains the HostAP struct ieee80211_mgmt.

The struct rule_test holds a rule test. It is:

```
struct rule_test{
    int test;            /* test keyword integer value */
    char test_data1[TEST_DATA_SIZE]; /* additional info for test */
    struct rule_test *next;       /* next test for this rule */
};
```

The struct rule_node holds a rule. It is:

```
struct rule_node{
    int protocol;        /* only 802.11 has been implemented */
    int type_subtype;    /* 802.11 type/subtype */
    struct rule_test test_struct; /* rule test */
    int action;          /* action to take if test result is true */
    char action_data1[ACTION_DATA_SIZE]; /* action description */
    struct rule_node *prev; /* previous rule node */
    struct rule_node *next; /* next rule node */
};
```
The type-subtype field in rule node can contain a type only, management, data, or control, or a subtype such as authentication, which is a subtype of type management, or data_null, which is a subtype of type data. If the type-subtype field contains a type only, then the rule is tested for all subtypes of the specified type. For example, if the type-subtype field contains management then the rule will test any 802.11 header with a management subtype such as association request, association response, disassociation, authentication. If the type-subtype field contains a subtype such as authentication, the rule will test only the 802.11 frame with the authentication subtype. The type-subtype field stores the hex value of the type and subtype as defined in the IEEE standard for 802.11.

Some examples of rule type-subtypes are [24]:

<table>
<thead>
<tr>
<th>type-subtype</th>
<th>hex</th>
<th>binary value</th>
<th>subtype</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyword</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>define MGMT_ASREQ</td>
<td>0x0</td>
<td>/* 0000 00</td>
<td>Association Req */</td>
<td></td>
<td></td>
</tr>
<tr>
<td>define MGMT_ASRES</td>
<td>0x10</td>
<td>/* 0001 00</td>
<td>Association Res */</td>
<td></td>
<td></td>
</tr>
<tr>
<td>define MGMT_REREQ</td>
<td>0x20</td>
<td>/* 0010 00</td>
<td>Reassoc. Req. */</td>
<td></td>
<td></td>
</tr>
<tr>
<td>define MGMT_RERES</td>
<td>0x30</td>
<td>/* 0011 00</td>
<td>Reassoc. Resp. */</td>
<td></td>
<td></td>
</tr>
<tr>
<td>define CONT_PS</td>
<td>0xa4</td>
<td>/* 1010 01</td>
<td>Power Save */</td>
<td></td>
<td></td>
</tr>
<tr>
<td>define CONT_RTS</td>
<td>0xb4</td>
<td>/* 1011 01</td>
<td>Request to send */</td>
<td></td>
<td></td>
</tr>
<tr>
<td>define DATA_DATA</td>
<td>0x08</td>
<td>/* 0000 10</td>
<td>Data */</td>
<td></td>
<td></td>
</tr>
<tr>
<td>define DATA_DTCFACK</td>
<td>0x18</td>
<td>/* 0001 00</td>
<td>Data + CF-Ack */</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The rule_nodes are stored in a tree like data structure made up of linked lists (see figure 6). The struct rule_tree_node holds the rule nodes.

```c
struct rule_tree_node {
    int protocol;       /* only 802.11 has been implemented */
    int type_subtype;   /* 802.11 type/subtype */
    struct rule_tree_node *parent;  /* parent node */
    struct rule_tree_node *prev;    /* previous node */
    struct rule_tree_node *next;    /* next node */
    struct rule_tree_node **children;  /* child node */
    struct rule_node *rules;        /* list of rules */
};
```

Since only struct ieee80211_mgmt {...} is needed from the HostAP file ieee802_11.h, it is included in ruleslist.h directly rather than through the statement #include < ieee802_11.h>.
struct ieee80211_mgmt {
    u16 frame_control;
    u16 duration;
    u8 da[6];
    u8 sa[6];
    u8 bssid[6];
    u16 seq_ctrl;

    union {
        struct {
            u16 auth_alg;
            u16 auth_transaction;
            u16 status_code;
            /* possibly followed by Challenge text */
            u8 variable[0];
        } __attribute__((packed)) auth;
        struct {
            u16 reason_code;
        } __attribute__((packed)) deauth;
        struct {
            u16 capab_info;
            u16 listen_interval;
            /* followed by SSID and Supported rates */
            u8 variable[0];
        } __attribute__((packed)) assoc_req;
        struct {
            u16 capab_info;
            u16 status_code;
            u16 aid;
            /* followed by Supported rates */
            u8 variable[0];
        } __attribute__((packed)) assoc_resp, reassoc_resp;
        struct {
            u16 capab_info;
            u16 listen_interval;
            u8 current_ap[6];
            /* followed by SSID and Supported rates */
            u8 variable[0];
        } __attribute__((packed)) reassoc_req;
        struct {
            u16 reason_code;
        } __attribute__((packed)) disassoc;
    }
}
u8 timestamp[8];
u16 beacon_int;
u16 capab_info;
/* followed by some of SSID, Supported rates,
 * FH Params, DS Params, CF Params, IBSS Params, TIM */
u8 variable[0];
} __attribute__ ((packed))  beacon;
} u;
} __attribute__ ((packed)) ;

6.3.2.4 ruleslist.c

Ruleslist.c performs the intrusion detection work in HostAP. It opens HostAP access point log file; writes messages to the log file; processes rules; creates the rule nodes and stores them in the rules tree, performs tests for valid type-subtypes, tests test keywords, and action key words; and applies rules to 802.11 frames and performs the rule action if the rule tests criteria are met.

Important functions in ruleslist.c are described below.

6.3.2.4.1 int process(struct rule_node *rulePtr){…}

Rules are received from the controller and processed by the function process(...) and stored in a rule_node struct. Rules longer than RULE_SIZE are truncated and only first RULE_SIZE characters are used. Process(...) calls functions to check type_subtypes against valid 802.11 types and subtypes, to check tests against valid tests, and to check actions against valid actions. If the rule is valid, the data is stored in the rule_node. If rule has invalid data, the rule is skipped and process(...) prints "Error in line x. Rule not used." where x is the line number of the bad rule in the rules file.

6.3.2.4.1.1 Tree description

The tree contains the intrusion detection rules used by WIDS. There are two branches under the root. The 802.11b branch can be used for wireless cards running in either the hostAP mode or monitor mode. The HFA384x branch requires a wireless card to be in monitor mode and needs the two card implementation of WIDS. Rules were implemented only for the 802.11b branch.
Figure 6: Rules Tree

6.3.2.4.1 int applyRules(struct ieee80211_mgmt *mgmt, hostapd *hapd){…}

Since the frame control structure in 802.11 frames is the same for management, data and control type-subtypes, the struct ieee80211_mgmt can be used to apply rules for all the type-subtypes.

When HostAP receives an 802.11 frame, the applyRules(…) function is called to apply the rules to the frame. ApplyRules() searches the rule tree for rules with the same type-subtype as the 802.11 frame. If a rule’s type-subtype matches the 802.11 frame control type-subtype the applyRules(…) calls the function

    int applyARule(struct rule_node *rule, struct ieee80211_mgmt *mgmt, hostapd *hapd){…}

applyARule calls
int tests(struct rule_node *rule, struct ieee80211_mgmt *mgmt, hostapd *hapd){…}

to perform the tests.

As noted above, a rule can have more than one test and all tests in a rule must return true for rule action to be taken. As soon as one test returns false, the rule is false and no other tests for that rule need to be performed. If tests(…) returns true, which indicates that all test criteria have been met, applyARule(…) calls int actions(struct rule_node *rule,struct ieee80211_mgmt *mgmt, hostapd *hapd, u8 addr[]){…} to perform the action specified by the rule. ApplyRules(…) continues searching the rule tree for rules with the same type-subtype as the 802.11 frame and repeats the above procedures for each applicable rule.

6.3.2.4.2 int tests(struct rule_node *rule, struct ieee80211_mgmt *mgmt, hostapd *hapd){…)

Tests(…) checks each rule for more than one test and determines the order of the tests to be performed. If a test needs data from an IP packet, it is tested last. Testing IP data requires forking a new process to listen for IP packets on wlan0. If any other test is false, testing the IP data and creating a new process can be ignored. Tests(…) compares the rule’s test data to the data in the applicable 802.11 frame field in accordance with the operation described in the test keyword and returns true or false.

If an intrusion detection rule needs info from an IP packet such as the time to live (TTL), HostAP forks a new process and runs a program scoop.c that listens for IP packets on wlan0. The IP listener program is based on libpcap and communicates with HostAP via shared memory. When an IP packet meeting the rule criteria is received, the IP listener signals HostAP and HostAP takes the action that is described in the intrusion detection rule.

The shared memory is initialized when the access point starts up. Semaphores are used to control access to the shared memory. The shared memory and semaphore id’s are stored in the hostapd struct. The new process that listens for IP packets needs to know the shared memory and semaphore id’s. They are passed to the new process as command line arguments in the execl command. The IP listener process, scoop [22], is started as follows:

    sprintf(filter,"%s%s","ip[8] > ", rule->test_data1);
    execl("/home/jvigo/BOSNST/Scoopshared/scoop","scoop", "-i", "wlan0","-a",ascshmid,"-m",ascsemid,"-x",filter,(char*)0);

A program named scoop was used for the IP packet listener. Scoop was modified to access the HostAP shared memory and to test the rules contained in the shared memory. Only TTL rule has been implemented.
Scoop uses the packet capture library libpcap and a filter for packet capturing can be given as a command line argument to scoop. All or part of the rule test information can be entered as a filter and then any packet captured by scoop has met the criteria of the test included in the filter. For example, if the test is for packets with TTL = 64, then HostAP sets the filter as "ip[8] = 64". TTL starts at position 8 in the IP header. The value of TTL is given by rule->test_data1. Any packet captured by scoop has TTL = 64.

On termination scoop detaches from shared memory segment and removes the shared memory segment and semaphore.

6.3.2.4.3 int flagtests(struct rule_test *rule, struct ieee80211_mgmt *mgmt, u16 flagtype){…}

Tests 802.11 header flags.

6.3.2.4.4 void iptestresult(hostapd *hapd){ …}

Iptestresult(…) is called to take rule action if IP packets are tested and rule tests are true.

6.3.2.4.5 int actions(struct rule_node *rule,struct ieee80211_mgmt *mgmt, hostapd *hapd, u8 addr[]){ … }

Actions (…) gives action to take if tests returned true.

6.3.2.4.6 Other functions

Ruleslist.c contains various other functions including functions to manipulate the rules tree, search the tree, find rules with a specific type or type_subtype, create new nodes, add nodes to rule tree, extract type from the 802.11 type_subtype value, find tree nodes with a specific type_subtype in a list, log messages, convert a MAC address from a character string to an unsigned integers and vice versa.

6.3.2.5 validmac.c

Validmac.c creates a hash table, fills it with key value pairs from a file, and provides functions to insert new items into the table and to find an item in the table with key. Validmac.c uses the library libtable.a from the program table-4.3.0 for the hash table and related functions for hash-table manipulations.

Hash tables are used for two things in this program:

1. to store the Organizationally Unique Identifier (OUI) numbers(keys) and manufacturers
(values) for MAC hardware addresses. If an OUI is not found in the table, then it is not
part of a valid MAC address. If an OUI is found in the table, the manufacture's name is
returned. The Organizationally Unique Identifier (OUI) can be obtained from
http://standards.ieee.org/regauth/oui/oui.txt. A perl utility program convert_ouihost.pl
was written to convert the oui.txt file to the format needed by this program.

2. to store the intrusion detection rules numerical values (keys) for type-subtypes, tests
keywords, and actions keywords and their text descriptions (values). In order to make the
screen messages and logs readable, descriptive text is used instead of the hex values. For
example, “type_subtype is Authentication” and “rule test is
SOURCE_MAC_NOT_EQUAL” are more understandable than “type_subtype is 0xb0”
and “rule test is 0x1f.” The file ruleslist.h contains the key value information needed. The
file names used as input for the hash tables are specified in the hostapd.conf file.

6.4 Other Programs Used in the WIDS System

The WIDS software uses several other programs that were modified to work with HostAP.

6.4.1 fakeap

Fakeap [25] creates fake 802.11b access points. It can be used as a rule action.

Fakeap was modified to read MAC addresses and vendors from file that contains authorized OUI
values for MAC address. Fakeap sends out an unlimited number of fake access points. When the
fakeap action is triggered, a window pops up on the controller’s screen that has a button that is
used to stop sending fake access points and to restore the WIDS AP original essid and MAC
address.

6.4.2 scoop

Scoop is written by Mike D. Schiffman [26]. Scoop is a program which captures and decodes IP
frames. Some changes were made to scoop to enable it use the intrusion detection rules.

The structs for sharedinfo, rule_node, and rule_test were added to scoop.h. Scoop.c was modified
to read the shared memory id and the semaphore id from command line arguments. The struct
sharedinfo *shared_ptr is set to the shared memory id.

A function void testpacket(struct scoop_pack *vp){…} was added to scoop.c to test other rule
criteria which is not filtered out. The TTL example above captures all packets with TTL = 64.
We may really want only packets with TTL = 64 from the MAC 00:02:2d:2e:5c:61.
testpacket(…) handles the source MAC equals 00:02:2d:2e:5c:61 test by comparing the MAC addresses included in the sharedinfo struct with the MAC in the ethernet frame received by scoop. If the two MAC’s are same, scoop sends a SIGUSR2 to the HostAP process to notify it that the rules criteria are true. When HostAP receives the SIGUSR2 message, it takes the action included in the rule.

6.4.3 table-4.3.0

Table [27] is a library for an open hash table with buckets and linked lists and functions to manipulate the table data.

6.5 WIDS Controller

Two of the controller’s responsibilities are to provide the access points with information needed to configure/reconfigure the directional antenna and to provide the intrusion detection rules for the access point.

The controller reads a configuration file at startup. The configuration file contains a hash of HostAP/WIDS access point names and their beginning sweep positions, ending sweep positions, and report rates; a hash of internal (non-WIDS) access points and their locations; the size of the grid for specifying access point locations; and a hash of the HostAP/WIDS access point names and their related locations, a numerical ID, and the midpoint of the beginning and ending sweep positions. The access points ID’s are numbers starting with 1 for left bottom corner of the perimeter and proceeding in a counterclockwise direction, 2 for the next access point, 3 for the next access point, etc. until all access points have a number. Numbering in this matter eliminates the need to consider x and y position values when comparing the location of one access point compared to the location of another.

After the controller configuration is completed, the controller displays a GUI containing an image of the area to be protected and the internal non-WIDS access points. The controller opens a socket on a specified port and listens for messages from the access points. Messages are handled as discussed in the previously described protocol.

WIDS access points are displayed on the GUI as they are connected to the controller. The position of the antenna of each access point is designated by a sweeping line. The controller assumes that a station associates with the access point closest to it.

The controller operator can change the rules file, display statistics about the stations connected to the WIDS AP or query the log file. Screen shots of those operations are shown below.
Figure 7: Change rules file
Figure 8: Show Stats
Figure 9: WIDS log
The controller uses a simple algorithm to determine which access point to reconfigure when a maximum signal strength position has been received from an access point. If the position is greater than the access point's sweep midpoint, then the access point with the next highest ID number is reconfigured. If the position is less than the sweep midpoint, the access point with the next lowest ID number is reconfigured. When maximum signal strength positions for a MAC address are received from two WIDS access points, the controller can locate the access point's position using trigonometry (the law of sines and Pythagoras' theorem). The GUI draws a red box at the calculated location of the intruder.
6.6 WIDS Simulation

6.6.1 Overview

The WIDS system can be run with simulated access points and simulated stations for testing or presentation purposes. The simulation handles up to sixteen WIDS access points. The WIDS access points can be either real (non-simulated) WIDS/HostAP access points, simulated access points or a combination of the two. If real WIDS/HostAP access points are used in the simulation, real stations can associate with them. The simulation does not handle simulated stations associating with a non-simulated WIDS/HostAP access point.

The simulation is written in Perl-TK. The simulation can be run on the same computer as the controller, the same computer as a real WIDS/HostAP access point, or a separate computer. It can be run from a command line or from a GUI. Use of the GUI requires a configuration file giving geometry for locating windows and a hash of commands for starting access points, controller, and virtual station. Other information in the simulation configuration file is controller IP and port for listening, access point name, access point port, delay per degree value, location, name of file containing simulated antenna readings, and an association allowed value. The delay per degree value determines the sweep speed of the simulated antenna. The association allowed determines if clients can associate with the access point.

If the simulation is run on the same computer as the controller or a real WIDS/HostAP access points, the performance of the WIDS system may decline significantly.

Figure 11: WIDS Simulation Control GUI
6.6.2 WIDS simulated access point

A WIDS simulated access point associates with a real (non-simulated) controller and receives no 802.11b frames. A simulated access point has all the functionality of a real WIDS/HostAP access point except it doesn't process or test rules, the only 802.11b type/subtypes implemented are associate and disassociate, and it doesn't compute the maximum signal strength location using least squares regression. It just picks the position of the maximum value signal strength as a position with POINTS/2 increasing trend on one side and POINTS/2 decreasing trend on other side.

6.6.3 WIDS simulated station

The simulated station configuration file contains a hash of MAC addresses, locations, begin seconds and end seconds, a list of IP:port pairs representing simulated access points, the time between associates, and the timeframe that a station attempts to associate. The timeframe is designated by BEGIN and END in the configuration file. A station appears at BEGIN seconds after the virtual station simulation is started and ends END seconds after the virtual station
simulation is started. The simulation sends ASSOCIATE messages to the list of IP:port pairs for simulated stations whose timeframe is in effect. The rate of sending the ASSOCIATE messages is determined by the time between associates value in the configuration file.

6.6.4 WIDS simulation - other

The original controller and WIDS simulation were written by Dr. Golden Richard III. I have extended the simulation and controller to include more features and to handle real access points as part of the work for this thesis.
Chapter 7: Tests and Tests Result

7.1 Test 1 08/25/04

7.1.1 Description of test

There are two neighboring access points connected to the controller. One is a real (non-simulated) WIDS AP access point, named HOSTAP, and the other is a simulated access point, named BETA, running on the same computer as the WIDS AP access point.

The real stations, with MAC address 00:02:2D:2E:5C:61 and MAC address 00:0D:88:4B:E0:D6, both associate with the real WIDS AP access point. These are real associations. They are not simulated. The rules are applied to the 802.11 frames received from both of the real stations. The WIDS AP access point computes the position of the station 00:02:2D:2E:5C:61 maximum single strength using the data in its simulated antenna readings file and sends it to the controller. The WIDS controller reconfigures the simulated access point sweep arch.

The simulated access point computes the position of the station 00:02:2D:2E:5C:61 maximum single strength using the data in its simulated antenna readings file and sends it to the controller.

The controller computes the position of the station 00:02:2D:2E:5C:61 from the data received from the two access points.

In order to test the IP packet reading rule, the stations 00:02:2D:2E:5C:61 and 00:0D:88:4B:E0:D6 ping the WIDS/HostAP access point. Four pings were sent from each station.

The station 00:0D:88:4B:E0:D6 is used only to test the rules. It does not appear on the WIDS controller GUI.

7.1.2 Expected results

The rule “MGMT_ASREQ,(FLAG_RETRY:OFF!SOURCE_MAC_EQUALS:00022D2E5C61!), LOG,FLAG RETRY Off” should be true. The controller log should show an entry for HOSTAP with a message including the date and time and “FLAG RETRY Off, Association Req.”

The rule “MGMT_AUTH,(SOURCE_MAC_EQUALS:00022D2E5C61!TTL_EQUALS:64!FLAG_RETRY:OFF!),LOG, 02 2d 2e 5c 61 authenticated with ttl = 64” should be true. The controller log should show an entry for HOSTAP with a message including the date and time and
“02 2d 2e 5c 61 authenticated with ttl = 64, Authentication.” Since four pings are sent with TTL = 64, the log should show four messages.

All other rules are false and no action should be taken.

The controller log should show association messages with dates and times for station 00:02:2d:2e:5c:61 with an Agere Systems wireless card and for station 00:0d:88:4b:e0:d6 with a D-Link wireless card.

The WIDS/HostAP log should show messages sent to the controller as a result of the rules.

HostAP logs all authentications and associations in the /var/log/messages log. There should be messages with times and dates for authentication and association for stations 00:02:2D:2E:5C:61 and 00:0d:88:4b:e0:d6.

Simulated antenna position of simulated access point BETA antenna when maximum signal received from station 00:02:2D:2E:5C:61 is 63 degrees. Simulated antenna position of HostAP access point HOSTAP antenna when maximum signal received from station 00:02:2D:2E:5C:61 is -25 degrees. The station is located at (452.94, 334.64).

Pings will be sent from 00:02:2D:2E:5C:61 and 00:0d:88:4b:e0:d6 with TTL = 64. HostAP and controller logs should show test results for IP headers from 00:02:2D:2E:5C:61 and not from 00:0d:88:4b:e0:d6.

7.1.3 Results of test

The controller log shows only the two lines expected as results of the intrusion detection rules (times are gm time):

HOSTAP Thu Aug 26 02:48:13 2004 FLAG RETRY Off Association Req
HOSTAP Thu Aug 26 02:51:56 2004 02 2d 2e 5c 61 authenticated with ttl = 64 Authentication

The controller log shows the expected association messages from the two stations:

HOSTAP Thu Aug 26 02:48:16 2004 associated with 00:02:2d:2e:5c:61 wireless card Agere Systems
HOSTAP Thu Aug 26 02:48:20 2004 associated with 00:0d:88:4b:e0:d6 wireless card D:Link Corporation

The WIDS/HostAP log shows messages:

,FLAG RETRY Off,Association Req
The HostAP access point /var/log/messages log shows authentication and association of stations 00:02:2d:2e:5c:61 and 00:0d:88:4b:e0:d6.

The controller received the antenna position of 63 degrees for simulated access point BETA antenna when the maximum signal message was received from station 00:02:2D:2E:5C:61 and the antenna position of -25 degrees for HostAP access point HOSTAP antenna when maximum signal received message was received from station 00:02:2D:2E:5C:61:

WIDS controller received client located at -25 (angle) from HOSTAP
WIDS controller received client located at 63 (angle) from BETA

The controller computed the station’s location as (452.94, 334.64):

ALERT client found at x = 452.936474021111 y = 334.641469930032

HostAP and controller logs show test results for IP headers from 00:02:2D:2E:5C:61 and not from 00:0d:88:4b:e0:d6. As additional testing on the IP packets, Scoop was run with the print hex option to dump the Ethernet and IP headers. The hostAP terminal window displayed the following for pings from 00:02:2D:2E:5C:61 and pings from 00:0d:88:4b:e0:d6. Only the ping from 00:02:2D:2E:5C:61 shows a test result as expected.

```
IP: 192.168.0.69 -> 192.168.0.70 (60) id: 15773 ICMP: echo reply
  00 0002 2d2e 5c61 0004 5a0c d80c 0800 4500
  10 003c 3d9d 0000 4001 bb48 c0a8 0045 c0a8
  20 0046 0000 535c 0100 0100 6162 6364 6566
  30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
  40 7761 6263 6465 6667 6869
scoop testing packet
  test result is ....
```
00 0002 2d2e 5c61 0004 5a0c d80c 0800 4500
10 003c 3d9e 0000 4001 bb47 c0a8 0045 c0a8 ...

IP: 192.168.0.69 -> 192.168.0.71 (60) id: 3257 ICMP: echo reply
00 000d 884b e0d6 0004 5a0c d80c 0800 4500
10 003c 0cb9 0000 4001 ec2b c0a8 0045 c0a8
20 0047 0000 535c 0100 0100 6162 6364 6566
30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
40 7761 6263 6465 6669 6869
scoop testing packet
IP: 192.168.0.69 -> 192.168.0.71 (60) id: 3258 ICMP: echo reply
00 000d 884b e0d6 0004 5a0c d80c 0800 4500
10 003c 0cb9 0000 4001 ec2a c0a8 0045 c0a8 ...

7.1.4 Test Results Conclusion

The test produced the expected results.

7.1.5 Screen Shot of Test 1

![Screen Shot of Test 1](image)

Figure 13: Reconfigured BETA AP and HOSTAP locating station
7.1.6 Hardware used

One IBM ThinkPad Pentium III 450 MHz, 128 Mb memory laptop with a Linksys Instant Wireless WPC11 wireless card (MAC address 00:04:5a:0c:d8:0c) running Linux and HostAP used for a WIDS/HostAP access point.

One generic Pentium 166 MHz, 32 Mb memory laptop with an Lucent Technologies Orinoco Silver 11 Mbits/s wireless card (MAC address 00:02:2D:2E:5C:61) address running Windows98 used for a station connecting to the WIDS/HostAP access point.

One generic Pentium 166 MHz, 64 Mb memory pc, with a D-Link 520 vE1 wireless card (MAC address 00:0d:88:4b:e0:d6) running Windows98 used for a station connecting to the WIDS/HostAP access point.

One Dell Pentium III 500 MHz, 192 Mb memory pc running Linux used for the controller.

One Soho 8 port 10Mbps Ethernet hub connecting the WIDS/HostAP access points and the controller.

7.1.7 Test Data

7.1.7.1 Rules file for test

```
MGMT_ASREQ,(FLAG_RETRY:OFF!SOURCE_MAC_EQUALS:00022D2E5C61!),LOG,FLAG_RETRY Off
MGMT_ASREQ,(FLAG_RETRY:ON!SOURCE_MAC_EQUALS:00022D2E5C61!),LOG,FLAG_RETRY On
MGMT_ASREQ,(DEST_MAC_EQUALS:00022D2E5C61!),LOG,assoc request dest mac 00 02 2d 2e 5c 61
MGMT_AUTH,(SOURCE_MAC_EQUALS:00022D2E5C61!TTL_EQUALS:64!FLAG_RETRY:OFF!),LOG,02 2d 2e 5c 61 authenticated with ttl = 64
#MGMT_AUTH,(SOURCE_MAC_EQUALS:000D884BE0D6!TTL_EQUALS:64!FLAG_RETRY:OFF!),LOG,00:0d:88:4b:e0:d6 authenticated with ttl = 64
#MGMT_AUTH,(SOURCE_MAC_EQUALS:00022D2E5C61!TTL_EQUALS:64!FLAG_RETRY:OFF!),DISASSOC, 02 2d 2e 5c 61 disassociated with ttl = 64
```

7.1.7.2 Access point MAC address 00:02:2D:2E:5C:61 antenna readings file

filename "angle4"

# format for a reading is degrees,signal strength;mac address,
# first number represents the angle in degrees. The second number represents
# the signal strength.
# next 6 numbers represent mac address. Each reading must be separated by ',',
# or '/n' and there must be a '\n' before the EOF character
# comments start with '#' and the '#' is allowed only at start of line
-35,10;00:02:2D:2E:5C:61,-34,12;00:02:2D:2E:5C:61,-33,16;00:02:2D:2E:5C:61,-
7.1.7.3 Simulated access point antenna readings file

filename "data8"

57,32;00:02:2D:2E:5C:61
58,33;00:02:2D:2E:5C:61
59,38;00:02:2D:2E:5C:61
60,40;00:02:2D:2E:5C:61
61,44;00:02:2D:2E:5C:61
62,45;00:02:2D:2E:5C:61
63,47;00:02:2D:2E:5C:61
64,44;00:02:2D:2E:5C:61
65,43;00:02:2D:2E:5C:61
66,41;00:02:2D:2E:5C:61
67,30;00:02:2D:2E:5C:61
68,29;00:02:2D:2E:5C:61

7.1.8 Details of test results

7.1.8.1 HostAP log


HOSTAP HOSTAPWed Aug 25 21:51:56 2004 , 02 2d 2e 5c 61 authenticated with ttl = 64,Authentication
Wed Aug 25 21:51:57 2004 , 02 2d 2e 5c 61 authenticated with ttl = 64,Authentication
Wed Aug 25 21:51:58 2004 , 02 2d 2e 5c 61 authenticated with ttl = 64,Authentication
Wed Aug 25 21:51:59 2004 , 02 2d 2e 5c 61 authenticated with ttl = 64,Authentication

7.1.8.2 Controller log

hostap.log opened gmtime Thu Aug 26 03:21:49 2004
HOSTAP Thu Aug 26 02:48:13 2004 FLAG RETRY Off Association Req
HOSTAP Thu Aug 26 02:48:16 2004 associated with 00:02:2d:2e:5c:61 wireless card Agere Systems
HOSTAP Thu Aug 26 02:48:20 2004 associated with 00:0d:88:4b:e0:d6 wireless card D:Link Corporation
HOSTAP Thu Aug 26 02:51:56 2004 02 2d 2e 5c 61 authenticated with ttl = 64
Authentication
HOSTAP Thu Aug 26 02:51:57 2004 02 2d 2e 5c 61 authenticated with ttl = 64
Authentication
HOSTAP Thu Aug 26 02:51:58 2004 02 2d 2e 5c 61 authenticated with ttl = 64
Authentication
HOSTAP Thu Aug 26 02:51:59 2004 02 2d 2e 5c 61 authenticated with ttl = 64
Authentication

7.1.8.3 Controller Terminal display

dashipushi:/home/jv/widsfinal1 # ./controllernet
WIDS controller initializing using file "CONFIG.controllerhap".
KNOWN ACCESS POINT LIST:
CTR
CBR
PINKY
CTL
CBL
LAP1
BAP1
BAP3
ALPHA
BAP2
BETA
HOSTAP
TAP2
LAP3
LAP2
TAP1
WIDS controller configuration complete.
WIDS controller bringing up GUI.
WIDS controller: Image is 600 x 416.
WIDS controller listening on port 7777.
WIDS controller got connection.
WIDS controller sending CONFIG to BETA.
WIDS controller configured BETA, located @ (420,270).
WIDS controller got connection.
WIDS controller sending CONFIG to HOSTAP.
WIDS controller configured HOSTAP, located @ (420,350).
WIDS controller logging HOSTAP Thu Aug 26 02:48:13 2004 FLAG RETRY Off
Association Req
WIDS controller logging HOSTAP Thu Aug 26 02:48:16 2004 associated with
00:02:2d:2e:5c:61 wireless card Agere Systems

WIDS controller logging HOSTAP Thu Aug 26 02:48:20 2004 associated with
00:0d:88:4b:e0:d6 wireless card D:Link Corporation

WIDS controller received client located at -25 (angle) from HOSTAP
adding to hash 00:02:2D:2E:5C:61 HOSTAP;-25;1
SENDING RECONFIG to BETA POS = -25 -45 90 2
WIDS controller received client located at 63 (angle) from BETA
locating ap
name1 = BETA, position1 = 63, name2 = HOSTAP, position2 = -25, ap1x = 420, aply = 270
ALERT client found at x = 452.936474021111 y = 334.641469930032
WIDS controller received client located at -25 (angle) from HOSTAP
WIDS controller logging HOSTAP Thu Aug 26 02:51:56 2004 02 2d 2e 5c 61
authenticated with ttl = 64 Authentication
WIDS controller logging HOSTAP Thu Aug 26 02:51:57 2004 02 2d 2e 5c 61
authenticated with ttl = 64 Authentication
WIDS controller logging HOSTAP Thu Aug 26 02:51:58 2004 02 2d 2e 5c 61
authenticated with ttl = 64 Authentication
WIDS controller logging HOSTAP Thu Aug 26 02:51:59 2004 02 2d 2e 5c 61
authenticated with ttl = 64 Authentication
WIDS controller received client located at -25 (angle) from HOSTAP
Use of uninitialized value in concatenation (.) or string at wids.pl line 466,
<RULES> line 6.
mesg not defined type =
** OTHER SIDE HUNG UP 11111 **
WIDS controller received client located at -25 (angle) from HOSTAP
Use of uninitialized value in concatenation (.) or string at wids.pl line 466,
<RULES> line 6.
mesg not defined type =
** OTHER SIDE HUNG UP 11111 **
dashipushi:/home/jv/widsfinal1 #

The “use of uninitialized value …” and “mesg not defined type = “ comments are results of the
access points shutting down

7.1.8.4 HostAP terminal

Script started on Wed Aug 25 21:25:54
2004)0;root@localhost:/home/jvigo/hostap-0.0.3/hostapd2new_shared4_clean1

[root@localhost hostapd2new_shared4_clean1]# ./hostapd hostapddashipushi.conf
Configuration file: hostapddashipushi.conf
table_alloc done
table_alloc rule done
could not insert in table: key exists and no overwrite, key=00:01:C8, c= 915
could not insert in table: key exists and no overwrite, key=08:00:30, c= 14393
could not insert in table: key exists and no overwrite, key=08:00:30, c= 14395

table Alloc done
table Alloc rule done

HOSTAP;17;log opened ;Thu Aug 26 02:47:24 2004
ip add= 192.168.1.4, port 7777
WIDS AP HOSTAP connecting to CONTROLLER @ 192.168.1.4:7777.
controller registered sockfd= 4
hostap AP HOSTAP sending initial HELLO to WIDS controller.
WIDS hostap HOSTAP waiting for CONFIG message.
message = CONTROLLER;2;- 45;45;2;MGMT_ASREQ,(FLAG_RETRY:OFF!SOURCE_MAC_EQUALS:00022D2E5C61!),LOG,FLAG RETRY Off
MGMT_ASREQ,(FLAG_RETRY:ON!SOURCE_MAC_EQUALS:00022D2E5C61!),LOG,FLAG RETRY On
MGMT_ASREQ,(DEST_MAC_EQUALS:00022D2E5C61!),LOG,assoc request dest mac 00 02 2d 2e 5c 61
MGMT_AUTH,(SOURCE_MAC_EQUALS:00022D2E5C61!TTL_EQUALS:64!FLAG_RETRY:OFF!),LOG, 02 2d 2e 5c 61 authenticated with ttl = 64
#MGMT_AUTH,(SOURCE_MAC_EQUALS:000D884BE0D6!TTL_EQUALS:64!FLAG_RETRY:OFF!),LOG,00:0d:88:4b:e0:d6 authenticated with ttl = 64
#MGMT_AUTH,(SOURCE_MAC_EQUALS:00022D2E5C61!TTL_EQUALS:64!FLAG_RETRY:OFF!),DISASSOC, 02 2d 2e 5c 61 disassociated with ttl = 64

hostap AP HOSTAP received CONFIG: sweep -45 - 45, report rate 2 processing rules
Using interface wlan0ap with hwaddr 00:04:5a:0c:d8:0c and ssid 'jv.cs.uno.edu'
Flushing old station entries
Deauthenticate all stations
ieee802_11.c 10 /nieee802_11.c auth failed
wlan0: STA 00:02:2d:2e:5c:61 IEEE 802.11: authenticated
do auth stuff here
,FLAG RETRY Off,Association Req

log action FLAG RETRY Off

wlan0: STA 00:02:2d:2e:5c:61 IEEE 802.11: associated (aid 1)
do associate stuff here
get ant readings
simulated antenna readings ok
Modified Scoop 1.0 [IP packet sniffing tool]
<ctrl-c> to quit
scoop pid = 1304

HOSTAP AP HOSTAP got ASSOCIATE from 00:02:2d:2e:5c:61, signal=120 @ position unknown
HOSTAPieee802_11.c 10 /nieee802_11.c auth failed
wlan0: STA 00:0d:88:4b:e0:d6 IEEE 802.11: authenticated
do auth stuff here
ieee802_11.c 10 /nieee802_11.c auth failed
wlan0: STA 00:0d:88:4b:e0:d6 IEEE 802.11: authenticated
do auth stuff here
wlan0: STA 00:0d:88:4b:e0:d6 IEEE 802.11: associated (aid 2)
do associate stuff here
get ant readings
simulated antenna readings ok
HOSTAP AP HOSTAP got ASSOCIATE from 00:0d:88:4b:e0:d6, signal=0 @ position unknown
HOSTAPmax signal is at -25.6176 sig= 29
max signal is at -25.6176 sig= 29
IP: 192.168.0.69 -> 192.168.0.70 (60) id: 15773 ICMP: echo reply
00 0002 2d2e 5c61 0004 5a0c d80c 0800 4500
10 003c 3d9d 0000 4001 bb48 c0a8 0045 c0a8
20 0046 0000 535c 0100 0100 6162 6364 6566
30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
scoop testing packet

test result is ....
, 02 2d 2e 5c 61 authenticated with ttl = 64,Authentication
log action 02 2d 2e 5c 61 authenticated with ttl = 64
IP: 192.168.0.69 -> 192.168.0.70 (60) id: 15774 ICMP: echo reply
00 0002 2d2e 5c61 0004 5a0c d80c 0800 4500
10 003c 3d9e 0000 4001 bb47 c0a8 0045 c0a8
20 0046 0000 525c 0100 0200 6162 6364 6566
30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
40 7761 6263 6465 6667 6869
scoop testing packet

test result is ....
, 02 2d 2e 5c 61 authenticated with ttl = 64,Authentication
log action 02 2d 2e 5c 61 authenticated with ttl = 64
IP: 192.168.0.69 -> 192.168.0.70 (60) id: 15775 ICMP: echo reply
00 0002 2d2e 5c61 0004 5a0c d80c 0800 4500
10 003c 3d9f 0000 4001 bb46 c0a8 0045 c0a8
20 0046 0000 515c 0100 0300 6162 6364 6566
30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
40 7761 6263 6465 6667 6869
scoop testing packet

test result is ....
, 02 2d 2e 5c 61 authenticated with ttl = 64,Authentication
log action 02 2d 2e 5c 61 authenticated with ttl = 64
IP: 192.168.0.69 -> 192.168.0.71 (60) id: 3257 ICMP: echo reply
00 000d 884b e0d6 0004 5a0c d80c 0800 4500
10 003c 0cb9 0000 4001 ec2b c0a8 0045 c0a8
20 0047 0000 535c 0100 0400 6162 6364 6566
30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
40 7761 6263 6465 6667 6869
scoop testing packet

test result is ....
, 02 2d 2e 5c 61 authenticated with ttl = 64,Authentication
log action 02 2d 2e 5c 61 authenticated with ttl = 64
IP: 192.168.0.69 -> 192.168.0.71 (60) id: 3258 ICMP: echo reply
00 000d 884b e0d6 0004 5a0c d80c 0800 4500
10 003c 0cba 0000 4001 ec2a c0a8 0045 c0a8
20 0047 0000 525c 0100 0200 6162 6364 6566
30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
40 7761 6263 6465 6667 6869
scoop testing packet

scoop testing packet
IP: 192.168.0.69 -> 192.168.0.71 (60) id: 3259 ICMP: echo reply
00 000d 884b e0d6 0004 5a0c d80c 0800 4500
10 003c 0cbb 0000 4001 ec29 c0a8 0045 c0a8
20 0047 0000 515c 0100 0300 6162 6364 6566
30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
40 7761 6263 6465 6667 6869
scoop testing packet
IP: 192.168.0.69 -> 192.168.0.71 (60) id: 3260 ICMP: echo reply
00 000d 884b e0d6 0004 5a0c d80c 0800 4500
10 003c 0cbb 0000 4001 ec29 c0a8 0045 c0a8
20 0047 0000 505c 0100 0400 6162 6364 6566
30 6768 696a 6b6c 6d6e 6f70 7172 7374 7576
40 7761 6263 6465 6667 6869
scoop testing packet
max signal is at -25.6176 sig= 29
max signal is at -25.6176 sig= 29
handle term
killing 1304
cleaning up shared memory struct
eloop.c after loop
Removing station 00:0d:88:4b:e0:d6
Removing station 00:02:2d:2e:5c:61
Flushing old station entries
Deauthenticate all stations
]0;root@localhost:/home/jvigo/hostap-0.0.3/hostapd2new_shared4_clean1
[root@localhost hostapd2new_shared4_clean1]#
[root@localhost hostapd2new_shared4_clean1]# Script done on Wed Aug 25 21:54:12 2004

7.1.8.5 BETA terminal

[root@localhost root]# cd /root/wids
[root@localhost wids]# perl accesspt2net
WIDS AP BETA connecting to controller @ 192.168.1.4:7777.
WIDS AP BETA sending initial HELLO to WIDS controller.
WIDS AP BETA waiting for CONFIG message.
57,32;00:02:2d:2e:5c:61
adding to allowed mac hash 00:02:2d:2e:5c:61 associate allowed = 1
58,33;00:02:2d:2e:5c:61
59,38;00:02:2d:2e:5c:61
60,40;00:02:2d:2e:5c:61
61,44;00:02:2d:2e:5c:61
62,45;00:02:2d:2e:5c:61
63,47;00:02:2d:2e:5c:61
64,44;00:02:2d:2e:5c:61
65,43;00:02:2d:2e:5c:61
66,41;00:02:2d:2e:5c:61
67,30;00:02:2d:2e:5c:61
68,29;00:02:2d:2e:5c:61
WIDS AP BETA received CONFIG: sweep -45 - 45, report rate 2.
WIDS AP BETA listening on port 7001 for virtual client.
reconfig message -45 90 2
ap got reconfig
vcexists 0
reconf received
position -11, begin -45, end 90 at receive reconf in nfound
sending position  63, 00:02:2D:2E:5C:61, 0
pos =-45 end = 90 beg = -45
back to original config BETA count = 2 end =45 begin = -45 pos =-45, dir= 1
mac
0
[root@localhost wids]#
[root@localhost wids]#

7.1.9 Commands to run test

On HostAP access point laptop:

1. Start simulated access point

   /root/wids # perl accesspt2net

2. In another terminal window - start hostapd

   /home/jvigo/hostap-0.0.3/hostapd2new_sharedclean1 # ./hostapd
   hostapddashipushi.conf

On controller pc:

1. Start controller

   /home/jv/widsfinal1# ./controllernet

7.2 Additional comments

The test used one real WIDS access point and one simulated WIDS access point, a controller which handles both real and simulated WIDS access points, and two real 802.11b stations. The test produced the expected results and shows that the WIDS software successfully implements parts of the complete WIDS package. However, the software needs to be extended to handle the rotating directional antennas and tests of the location calculations and antenna reconfiguration need to be performed with the actual hardware.

The terminals show a lot of messages and frame data that are used for debugging purposes. Once
the software is sufficiently tested and debugged, most of those messages will be turned off and only essential messages will be displayed.
Chapter 8: Conclusion and suggestions for future work

8.1 Conclusion

This paper presented a description of the IEEE802.11b standard and some of the problems of 802.11b wireless networks which leave them vulnerable to attacks from unauthorized users. Several types of attacks were discussed and a Wireless Intrusion Detection System (WIDS) was proposed to improve the security of wireless networks. The WIDS uses access points equipped with rotating directional antennas to identify and respond to potential threats and to locate potential intruders.

The WIDS will surround an omni-directional wireless network and a remote station beyond the perimeter of the omni network will connect to a WIDS AP before connecting to the omni AP. A set of rules is used to determine whether or not the remote station is an intruder. The WIDS APs are connected to a controlling computer via a wired network. The controlling computer configures the WIDS AP’s and locates intruder stations using data from two WIDS APs.

The WIDS APs run HostAP software that is modified to include a rule based intrusion detection system, a protocol for communicating with a controlling computer via a wired network, and to process data from a simulated directional antenna. A laptop computer with a Prism wireless card is used as a WIDS AP. The controller computer software is written in Perl-TK. A combination of simulated and non-simulated tests was used to show that the WIDS can provide additions to 802.11b security. The WIDS controller is able to connect to real and virtual WIDS APs and real 802.11b stations are able to connect to a real WIDS AP. The antenna functions of a WIDS AP are simulated due the lack of rotating directional antennas for the WIDS APs. Simulated WIDS APs and stations are used to show how WIDS would operate with several WIDS APs and stations.

Although the WIDS APs were not fully implemented, this paper shows that it is feasible to provide additional 802.11b security using off the shelf hardware and open source software. Other research mentioned in the paper shows that it is possible to locate wireless stations with directional antennas. All that remains to complete a WIDS prototype is to build a WIDS AP with an embedded computer and a rotational antenna and to modify the WIDS HostAP software to control the movement of the antenna.

Even though IEEE801.11i improves security of wireless networks, there is still a need for additional security provided by WIDS. Installed wireless networks will not be immediately updated. New hardware is needed for CCMP, which could be expensive for large networks. Since IEEE 802.11i is new technology, there can be configuration or implementation errors. Authentication mechanisms can be compromised by human errors or by lost or stolen equipment.
8.2 Suggestions for further work

In addition to building a WIDS AP as mentioned above, there are several things that can be done to improve the WIDS implementation.

8.2.1 Expand rule processing

8.2.1.1. Handle logical OR in rules testing

Currently, all tests in a rule must be true for rule to be true. For a rule with three tests, this is equivalent to “test1 && test2 && test3” where testx is the result of a test. The rules processing could be modified to handle both the logical AND and OR operations and rule would test for “test1 || test2 || test3.”

8.2.1.2. Implement more IP rules and tests

The only field in an IP header that can be tested by a rule is the TTL field. The rule processing could be expanded to include tests of more IP header fields.

8.2.1.3 Implement HFA384 rx rules

Rules testing the radio information contained in the HFA384 Rx frames could be implemented. This requires a second wireless card in monitor mode to get time, signal, silence and rate.

8.2.2 Expand controller functions

8.2.2.1 Check for APs that are down

If the controller doesn’t receive reports from an AP for x seconds, it assumes the AP is down or it is a denial of service attack victim. The controller would reconfigure AP’s on each side of downed AP to sweep its territory and attempt to compute the location of a potential intruder.

8.2.2.2 Notify others of important events

The controller could “beep” and flash warning messages on its monitor when something important happens and it could send e-mail to other computers.

8.2.2.3 Improve handling of moving stations

A station’s location is computed once. If the station moves without reassociating, its new location is not computed. The controller could check for moving stations and recomputed the new locations.
References


http://www.securityfocus.com/news

http://www.securityfocus.com/news


http://www.commsdesign.com/showArticle.jhtml?articleID=16506047


[22] Lawrence L. Lapin, “Statistics for Modern Business Decisions”, Harcourt Brace Jovanovich, Inc. copyright 1973 page 496. (Some changes made to order of equations and variables were made
to fit solution into format of GSL. For example, regression equation in book is $a + bX + cX^2$, but GSL needs $aX^2 + bX + c$.


[25] Stuart Stock and Ken Beames, “fakeap”, Black Alchemy Weapons Lab. The version used is $Id: fakeap.pl,v 1.3 2002/08/31 20:56:42 shstock Exp $. Copyright (c) 2002 Black Alchemy Enterprises. All rights reserved.

[26] Information on scoop is $Id: scoop.c,v 1.2 2002/03/11 07:28:45 route Exp $ from book entitled “Building Open Source Network Security Tools”, scoop.c - Packet Sniffing Technique example code Copyright (c) 2002 Mike D. Schiffman <mike@infonexus.com> All rights reserved

Appendix

WIDS Access Point components

Stepper Motor Controller (STP 100)
http link: www.pontech.com/products/stp100/index.htm
Price: $159.00

Stepper Motor (Vexta PK268-01A)
http link: www.pontech.com/products/vexta/index.htm
Price: $99.00

Netgate Prism Card (NL-2511 CD PLUS EXT2)
http link: www.netgate.com/NL2511.html
Price: $75.00

Pigtail for the wireless card to allow connection of directional antenna
Price: $25.00

Celestron 93499 heavy duty tripod
http link: www.digitalfotoclub.com/products2/Celestron_Heavy_Duty_Tripod_93499.html
Price: $325.00

X-Scale Single Board Computer and Wireless Networking Platform (SP-KIT400, SPB400CA, SDC400CA)
http link: www.xbox.com/Products/Xscale.htm
Price: $795.00 (SP-KIT400)
    $575.00 (SPB400CA)
    $159.00 (SDC400CA)

Power supply +5V/15A, +12V/4A, -12V/1A, 24V/3A Supply, stock number 14066 PS
http link: www.powersupplydepot.com/14066-PS.htm
Price: $29.95

Directional antenna with mounting hardware
Price: $150.00

Additional supplies, including aluminum poles for mounting directional antennas, enclosures for the electronics, cables, solder, and electrical connectors
Price: $300.00
Vita

John Vigo was born in New Orleans, Louisiana. He graduated from the University of New Orleans with a B.S. degree in mathematics in 1975 and an M.S. degree in accounting in 1979. He passed the Certified Public Accounting exam in 1978 and worked for international public accounting firms in the United States and the Netherlands Antilles. He is currently employed as the accounting manager of a non-profit arts organization.