Enhancing the Security of Corporate Wi-Fi Networks Using DAIR

Paramvir Bahl[†], Ranveer Chandra[†], Jitendra Padhye[†], Lenin Ravindranath[†]
Manpreet Singh[‡], Alec Wolman[†], Brian Zill[†]
[†]Microsoft Research, [‡]Cornell University

ABSTRACT

We present a framework for monitoring enterprise wireless networks using desktop infrastructure. The framework is called DAIR, which is short for *Dense Array of Inexpensive Radios*. We demonstrate that the DAIR framework is useful for detecting rogue wireless devices (e.g., access points) attached to corporate networks, as well as for detecting Denial of Service attacks on Wi-Fi networks.

Prior proposals in this area include monitoring the network via a combination of access points (APs), mobile clients, and dedicated sensor nodes. We show that a dense deployment of sensors is necessary to effectively monitor Wi-Fi networks for certain types of threats, and one can not accomplish this using access points alone. An ordinary, single-radio AP can not monitor multiple channels effectively, without adversely impacting the associated clients. Moreover, we show that a typical deployment of access points is not sufficiently dense to detect the presence of rogue wireless devices. Due to power constraints, mobile devices can provide only limited assistance in monitoring wireless networks. Deploying a dense array of dedicated sensor nodes is an expensive proposition.

Our solution is based on two simple observations. First, in most enterprise environments, one finds plenty of desktop machines with good wired connectivity, and spare CPU and disk resources. Second, inexpensive USB-based wireless adapters are commonly available. By attaching these adapters to desktop machines, and dedicating the adapters to the task of monitoring the wireless network, we create a low cost management infrastructure.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management; Network monitoring*

General Terms

Management, Reliability, Security

Keywords

Wireless networks, 802.11, security, rogue AP, denial-of-service

1. INTRODUCTION

Many corporations make substantial investments in their wireless infrastructure. For example, Microsoft's IEEE 802.11 based wireless (Wi-Fi) network consists of approximately 5,000 access points (APs) supporting 25,000 users each day in 277 buildings, covering more than 17 million square feet [10]. In addition to the equipment costs, the costs of planning, deploying, and maintaining such networks is substantial. Thus, it is important to develop infrastructure that improves the ability of Information Technology (IT) departments to manage and secure their wireless networks.

In recent years, researchers have uncovered security vulnerabilities in Wi-Fi networks [20]. They showed that the Wired Equivalency Protocol (WEP), the popular 802.11 security mechanism that most corporations were using at the time, was fundamentally flawed. In a series of highly publicized papers, they showed that 802.11 networks could be compromised easily. The community reacted quickly by developing and deploying alternate security solutions including VPNs, IEEE 802.1x [30], several variations of EAP [14], Smart cards, and more recently WPA [29]. Yet, the wireless LAN (WLAN) security problem was not completely resolved.

Last year, Microsoft conducted a series of interviews with WLAN administrators of several large and small organizations [10]. The goal of these interviews was to understand the difficulties involved in deploying and managing corporate WLANs. The issue of WLAN security came up repeatedly during these interviews. All administrators felt that WLAN security was a problem. They were unhappy with the quality of the tools they had at their disposal. Many of them would periodically walk around their buildings using WLAN scanning software (e.g. NetStumbler [33]) looking for security vulnerabilities. Some hired expensive outside consultants to conduct security vulnerability analyses of their WLAN deployment, only to conclude that what they really needed was an on-going monitoring and alerting system. Most administrators believed that better systems to manage WLAN security are needed.

Even after protocols such as IEEE 802.1x and WPA are deployed, corporate networks can be compromised by off-the-shelf 802.11 hardware and software. For example, an unauthorized AP can be connected to the corporate Ethernet, allowing unauthorized clients to connect to the corporate network. The *rogue AP* may be connected by a malicious person or, as is more often the case, by an employee who innocently connects an AP in his office without realizing that he is compromising the corporate network. A rogue AP (or a *rogue ad-hoc network* [16]) can circumvent the elaborate security measures that the IT department may have put in place to protect the company's intellectual property.

To test our assertion that people inadvertently compromise the security of their networks, we conducted an experiment in two large organizations that had secured their WLANs using one of the methods mentioned previously. We walked around with a WLAN-enabled laptop in a small section of the two campuses looking for APs to which we could connect. Sure enough, we found several

MobiSys'06, June 19–22, 2006, Uppsala, Sweden.
Copyright 2006 ACM 1-59593-195-3/06/0006 ...$5.00.

"Rogue APs". We successfully connected to the rogue APs and we were able to browse the Internet and to access internal web servers in both organizations.

Beyond rogue APs and rogue ad-hoc networks, there are a number of other ways to attack corporate 802.11 networks. For example, *Eavesdropping*, where the attacker passively listens to the traffic on the wireless network and gleans useful information, *Denial of Service*, where an attacker exploits flaws in the 802.11 protocol to disable the wireless link and disrupt communication, *Phishing* (sometimes called Pharming), where the attacker impersonates a legitimate AP and lures unsuspecting clients to connect to it. Details of these and other attacks are provided in Section 2. The point is that WLAN security continues to be a challenge.

The effectiveness of any management solution for wireless networks depends upon the ability to perform RF sensing from a large number of physical locations. This is important both for coverage and for pinpointing the precise location of a problem. We designed the DAIR (*Dense Array of Inexpensive Radios*) system for building wireless network management applications that benefit from dense RF sensing. The virtues of our system and the different applications that we intend to build are described in our recent position paper [16]. In this paper, we describe the design, implementation and performance of our first wireless management application.

The DAIR approach is unique in that it builds on the following two important observations. First, in most enterprise environments one finds plenty of desktop machines. The machines are generally stationary and are connected to wall power. They have good wired connectivity, spare CPU cycles, free disk space, and high-speed USB ports. Second, inexpensive USB-based wireless adapters are readily available and their prices continue to fall.[1] By attaching USB-based wireless adapters to desktop machines, and dedicating the adapters to the task of monitoring the wireless network, we create a low-cost monitoring infrastructure that is then used to manage the security of the network.

The first DAIR application that we have built and deployed is an alert system that looks for security breaches in enterprise 802.11 networks. It correctly detects inadvertent security breaches by non-malicious users, and raises the bar against attacks by malicious users. It does not handle the case where a malicious user employs non-802.11 compliant wireless devices to connect to the network.

We have deployed the DAIR security management application in a *98 m* by *32 m* office building. Our current deployment uses 22 desktop machines. We have written 31,757 lines of C, C++, and C# code to build this system. The average CPU load on each of the desktop machines running the DAIR monitoring software is no more than 2.25%. The average CPU load on the DAIR server varies between 20 to 40% depending on the time of the day. Our server and desktop machines are older models with less CPU horsepower and memory than is typically available in current corporate systems. The additional network traffic due to DAIR is an insignificant 2.5Kbps from each desktop machine.

In summary, the primary contributions of this paper are:

- We provide specific examples of why standard authentication and encryption schemes are inadequate to secure corporate Wi-Fi networks, which motivates our solutions based on continuous monitoring of Wi-Fi networks.

- We show that to provide comprehensive coverage for detecting security breaches, a dense deployment of RF sensors is necessary.

- We describe how a scalable system of dense Wi-Fi sensors can be built inexpensively.

- We build such a system and evaluate its performance.

2. ATTACKS ON WI-FI NETWORKS

In this section, we describe some attacks that network administrators of corporate Wi-Fi networks have to guard against. We broadly classify these attacks as *passive* and *active*. The classification is important for understanding the strengths and limitations of the DAIR security management system.

Eavesdropping

Eavesdropping is a passive attack. The attacker passively listens to the traffic on the wireless network and gleans useful information. The listener may use sophisticated code breaking techniques [20]. Countermeasures include use of better encryption techniques as well as physical security measures such as use of radio-opaque wallpaper [5]. Passive attacks are difficult, if not impossible, to detect and we do not address them in this paper.

Intrusion

Any attack that allows a user to gain unauthorized access to the network is called an Intrusion attack. Intrusion attacks are active attacks and several such attacks are possible.

An attacker can compromise the corporate network by gaining physical access to its wired network and connecting a wireless AP to it. The AP creates a "hole" through which unauthorized clients can connect, bypassing the elaborate security measures that the IT department may have put in place. A similar attack can be carried out by using ad-hoc wireless networks instead of APs. A corporate network may also be compromised when an attacker finds and uses an unsecured AP connected to the network by an unsuspecting employee. The widespread availability of inexpensive, easy-to-deploy APs and wireless routers has exacerbated this problem. As mentioned earlier, we found several unsecured APs in large organizations. The DAIR security management system can detect both rogue APs and rogue ad-hoc networks. Another way a corporate network can be compromised is when an attacker obtains the credentials (e.g., WEP passwords, IEEE 802.1x certificates) needed to connect to the corporate network [20]. The DAIR security management system can not currently detect such attacks.

Denial of Service (DoS)

Denial of Service attacks are active attacks. A variety of DoS attacks are possible. Some DoS attacks exploit flaws in the IEEE 802.11 protocol. For example, a *disassociation attack* is where the attacker sends a series of fake disassociation or deauthentication messages, causing legitimate clients to disconnect from the AP [19]. In a *NAV attack*, the attacker generates packets with large duration values in the frame header, thereby forcing legitimate clients to wait for long periods of time before accessing the network [19]. In a *DIFS attack*, the attacker exploits certain timing-related features in the IEEE 802.11 protocol to aggressively steal bandwidth from legitimate users [34]. In all three cases, the attacker transmits packets in an abnormal way, either by generating non-compliant packets, or by transmitting compliant packets at an abnormally high rate. The DAIR security management system can detect such attacks. DoS attacks are also possible by creating large amount of RF noise in the neighborhood of the network. The DAIR security management system can detect such attacks by comparing current observations with historical data observed from multiple vantage points.

[1] On July 28th, 2005 at http://www.anandtech.com/, we found a sale price of $6.99 for an 802.11g USB adapter.

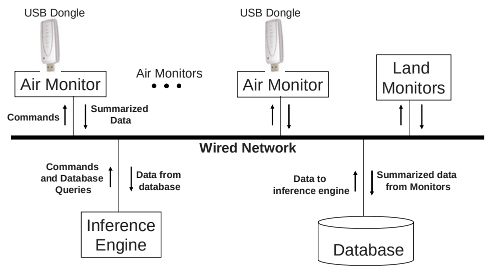


Figure 1: The DAIR Architecture.

DoS attacks can also be mounted by gaining access to the corporate wired network and attacking the APs from the wired side. The DAIR system does not handle DoS attacks on the wired network.

Phishing

Phishing is an active attack. An attacker sets up a wireless AP that masquerades as a legitimate corporate AP (same SSID, perhaps even same BSSIDs). If the client does not use mutual authentication, it is possible for the attacker to lure unsuspecting legitimate users to connect to its AP. The attacker can then use a variety of techniques to extract private information (for example, sniff for passwords). The DAIR system can detect phishing attacks. However, we do not describe solutions to phishing attacks in this paper.

3. DESIGN AND ARCHITECTURE

Figure 1 provides a high-level illustration of the major components of the DAIR system. In this section, we provide a detailed description of each of the components and describe the current status of our implementation.

The DAIR system is designed for easy deployment in enterprise environments, both large and small. DAIR makes use of existing enterprise desktop machines for monitoring. In such an environment, the IT department can decide which desktops will be used for monitoring, and they can also manage the process of deploying the DAIR software on such systems. Therefore, we expect that few incentives will be necessary to convince the primary users of these desktop computers to run the DAIR software on their machines. In a corporate environment, most users do not have administrative privileges to their desktop machines, so they will not be able to tamper with the DAIR software, either purposefully, or inadvertently. However, we must ensure that the DAIR monitoring software does not adversely impact the interactive performance of desktop computers it runs on.

3.1 The AirMonitors

We use the term AirMonitor to refer to an ordinary desktop computer in the enterprise that is equipped with an inexpensive USB 802.11 wireless card and has two components of the DAIR software installed: (1) the AirMonitor service; and (2) a custom device driver that works with any USB wireless card based on the Atheros chipset. The AirMonitor service is user-level code that runs as a Windows service, the equivalent of a daemon on Unix systems. The primary task of the AirMonitor is to listen continuously, either on a fixed channel, or in scan mode on a sequence of channels. The AirMonitor configures the wireless card in promiscuous mode, so that all 802.11 frames are received, including those destined for other 802.11 stations.

We modified the Windows device driver written by Atheros for their USB 802.11 chipset to support two new capabilities.[2] First, we added frame logging support to the driver so that all received 802.11 frames are copied into an in-kernel ring buffer. All frames are copied into this buffer, including those that have decoding errors – only those frames whose preamble cannot be decoded are discarded. Stored along with each frame is additional information about the frame reception, including the signal strength, the channel, and the data rate. We also added support to allow user-level programs to copy the contents of the kernel ring buffer, and to count how many frames are dropped if the ring buffer becomes full.

The other major capability we added to the driver is a new mode that we call "monitor mode." Monitor mode disables all of the driver's default scanning behavior. When the driver is not associated with a wireless network, it performs periodic active and passive scans. An active scan is performed by switching to each channel, issuing a probe request, and then waiting for probe responses from any surrounding access points. Passive scans are done by listening for beacons on each channel, in turn. Monitor mode is useful for two reasons: first, when monitor mode is enabled the AirMonitors become completely passive; second, when a particular channel is selected, the device will not automatically switch to other channels thereby missing some frames on the channel it was supposed to be monitoring.

The AirMonitor service contains all of the user-level code for monitoring. Figure 2 shows a diagram of the AirMonitor service internals. The AirMonitor service enables promiscuous mode, monitor mode, and frame logging at the driver level, at which point all frames are delivered to the service. Within the service, the basic unit of extensibility is a "filter": each new application built to use the DAIR system installs an application-specific filter that runs inside the AirMonitor service. The Filter Processor takes all frames from the driver and multicasts them to each running filter. The filter's primary task is to analyze the frames, summarize them in an application-specific manner, and then submit those summaries to the database server. For example, the filter that we use to assist with detecting rogue wireless networks summarizes all SSID's (network names) and BSSID's (Access Point MAC addresses) that it overhears,[3] and then submits those summaries to the database server every 90 seconds. To ease the task of building a new filter, the AirMonitor service contains a number of support modules. For example, filters make use of our 802.11 parser module to extract information from the frames, and they make use of our SQL helper module to assist with the task of submitting summaries to the database. The intent is that filters do whatever summarization is sensible to improve the scalability of the system without imposing an undue CPU burden on the AirMonitors – we don't want to submit every frame that each AirMonitor overhears to the database, yet we also don't want the AirMonitors to do all of the complex data analysis, which is the responsibility of the inference engine.

The Command Processor module of the AirMonitor service accepts commands from other components of the DAIR system (e.g., the DAIR management console, or one of the inference engines). Before accepting an incoming request, it checks to see if it can fulfill the request. For example, if an AirMonitor receives a new request to monitor a specific channel different from the one it is already monitoring, it will refuse that new request. Similarly, if the AirMonitor determines that the additional request will place un-

[2] We are not using the MADWiFi driver which supports similar functionality, but only for the Linux platform.

[3] Note that SSID/BSSID information is also available in frames other than beacons. Many 802.11 frames contain BSSIDs and probe responses contain both an SSID and BSSID.

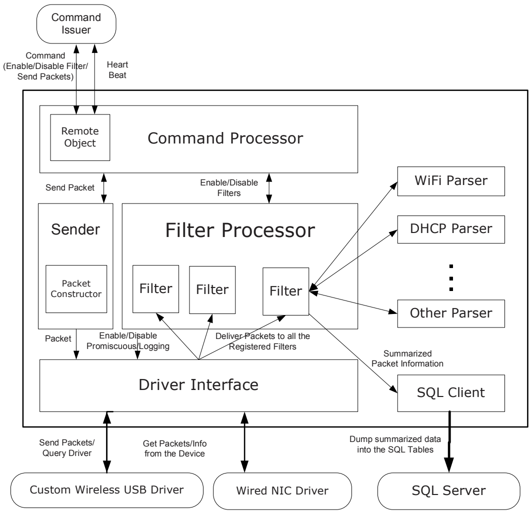

Command Issuer
Command (Enable/Disable Filter/ Send Packets)
Heart Beat
Remote Object
Command Processor
Send Packet
Enable/Disable Filters
WiFi Parser
Sender
Filter Processor
DHCP Parser
Packet Constructor
Filter
Filter
Filter
Other Parser
Packet
Enable/Disable Promiscuous/Logging
Deliver Packets to all the Registered Filters
Summarized Packet Information
Driver Interface
SQL Client
Send Packets/ Query Driver
Get Packets/Info from the Device
Dump summarized data into the SQL Tables
Custom Wireless USB Driver
Wired NIC Driver
SQL Server

Figure 2: The AirMonitor Architecture.

due burden on the host, it will refuse the request. While the precise definition of what constitutes undue burden varies based on circumstances, parameters such as history of CPU and memory usage are taken into consideration [24].

The AirMonitor nodes are not limited to passive observations. For example, an inference engine may request one of the AirMonitors to attempt to associate with an Access Point in order to gather more information. This requires the AirMonitor node to send association requests and to process incoming responses.

All of the components shown in the AirMonitor service diagram have been implemented. Furthermore, we currently have implemented four filters: one to summarize SSID and BSSID information for detecting rogue wireless networks, one to summarize disassociation frames, one to summarize data transfers between clients and access points, and one to summarize frames that appear to have abnormal duration (NAV) values.

3.2 The LandMonitors

Internally, the structure of the LandMonitors is identical to that of the AirMonitors. The key difference between the LandMonitor and the AirMonitor is that LandMonitors are used to monitor the wired network – as we will see in Section 4, two of our tests for detecting rogue wireless networks involve monitoring the wired network. We expect that LandMonitors will be deployed with much less density than AirMonitors, although this depends on the network configuration at a given site. For many organizations, having a single LandMonitor per subnet that continuously monitors the uplink to the subnet router will offer adequate visibility into the wired network. For those organizations that want greater visibility (e.g., for the replay test described in Section 4.1.1), many enterprise-class Ethernet switches can dynamically enable port mirroring, allowing a LandMonitor to cycle through many different links within a subnet. As with the AirMonitors, all the LandMonitor components have been implemented, and we have also implemented two filters for the LandMonitors: one for monitoring DHCP requests, and another to implement replay detection. The details of these filters are described in the next section.

3.3 The Inference Engine

The computationally intensive analysis tasks are all performed by the inference engines. As was the case with the filters in the AirMonitor service, each application that is built to run on the DAIR system installs an application-specific inferencing component that runs on one of the inference engine nodes. Our expectation is that IT administrators will allocate dedicated machines to inferencing rather than running these tasks on end-user's desktop computers.

The inference engines learn about new events by issuing periodic queries to the database server. For most applications, such queries only need to analyze data that was submitted to the database server by the AirMonitors after the previous query. To illustrate the kind of computation done by an inference engine, we briefly describe the inference engine for detecting rogue wireless networks. The inference engine issues periodic queries that look at all new arrivals in the "SSID and BSSID seen" table since the last query, and then checks whether any of those networks are not in the list of approved SSID's and BSSID's. If it finds an unknown network, then the inference engine issues commands to the AirMonitors to perform the sequence of tests used to decide whether the unknown wireless network is connected to the corporate wired network in question.

3.4 The Database

We use Microsoft's SQL Server 2005 as our database server. We made no custom modifications to the database server. Furthermore, apart from creating appropriate table layouts, indices and triggers, we did little to optimize the database performance. We plan to carry out further optimizations by using more sophisticated tools that perform workload-specific index tuning.

The DAIR system is designed to scale to handle very large enterprises. Our use of a centralized database does not limit the scale of the system because when the number of clients in the system exceeds the capacity of a single database server, one can simply deploy another database server. The only constraint is that clients should be assigned to servers in a location-aware manner, to limit the number of queries that must be performed across multiple database servers.

4. DETECTING ATTACKS

We now describe how we leverage the DAIR architecture to detect intrusion and denial of service attacks.

4.1 Intrusion Attacks

We focus on intrusion attacks that involve connection of unauthorized wireless equipment to a corporate network. There are many scenarios whereby rogue wireless equipment may be connected to a corporate network. For example, an employee might bring in a wireless AP from home and plug it in to the corporate network without configuring it to require the necessary authentication. Or a disgruntled employee may deliberately attach an unauthorized AP to the corporate network. Once an unauthorized AP is attached to the corporate network, the security of the network is compromised even if all the *authorized* APs are configured to use appropriate authentication mechanisms. Thus, detecting these unauthorized or "rogue" APs is an important challenge.

One may argue that the rogue AP problem is best solved by securing the wired network. For example, if the 802.1x protocol is deployed on the wired network, or if some form of MAC address filtering is employed, unauthorized access points will not be able to connect to the wired network. Similarly, VPN or IPSec based solutions can limit access to corporate resources to authorized clients. While these solutions are certainly useful, they do not fully solve the problem. An authorized client, connected to the wired network and equipped with a wireless interface, can bridge the two network interfaces to provide link-layer forwarding, or provide IP-level forwarding by acting as a NAT. The wireless interface can then be put in ad-hoc mode, and used to allow unauthorized clients to connect to the wired network. For example, Carnegie Mellon University has recently issued prohibitions [9] against having two active interfaces on the same machine. We posit that a distributed monitoring infrastructure, acting in addition to solutions like 802.1x and VPNs, provides a better solution to the problem.

It may appear at first glance that the monitoring infrastructure does not need to do much: an organization simply needs to maintain a database of all authorized APs, including their SSIDs and BSSIDs. An alarm is raised whenever an unknown SSID or BSSID is heard by a wireless sensor. This sensor can be an AP, a mobile client, or a dedicated sensor node. This is the basic mechanism proposed in previous research [15], and many wireless management companies offer rogue AP detection as part of their product offerings [1, 2]. Unfortunately, this straightforward approach is susceptible to both false negatives and false positives. We now discuss how the DAIR framework can be used to minimize both false positives and false negatives.

4.1.1 Guarding Against False Positives

In many office buildings, one is likely to overhear APs deployed by other corporations in the vicinity. The fact that an AirMonitor can hear an AP that is not in the database of authorized APs is not necessarily cause for alarm. The inference engine prioritizes the alarms by investigating whether the "suspect AP" (hereafter referred to as the *suspect*) is attached to the corporate network. If the inference engine determines that the suspect is indeed attached to the corporate network, then the alarm is assigned a higher priority. While it is not always possible to determine whether the suspect is connected to the corporate network, we have implemented several tests to answer the question in many situations.

Our tests for detecting whether the suspect is attached to the corporate network depend on detecting that the suspect device is forwarding packets between the wireless and the wired network.

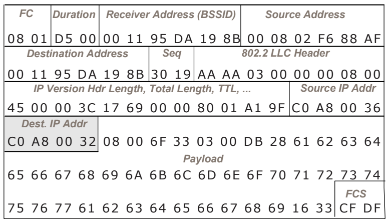
The suspect device can forward packets to the wired network in one of two ways. First, the suspect may forward frames at the link layer ("layer 2"), without involving higher layers of the networking stack. The term "access point", as defined in the 802.11 standard, refers to such a link layer forwarding device. Most commercial-grade access points are link-layer forwarders. Second, the suspect may forward packets at the IP layer, by acting as a router. Most wireless devices designed for home networking are IP-layer forwarders. These devices combine AP and router functionality, usually along with NAT capabilities.

We first describe a test that can reduce false positives regardless of whether the suspect is a link layer or IP forwarder. Then we describe a test that is useful when the suspect is a link layer forwarder. Finally we consider the case where the suspect is an IP forwarder.

Note that while we describe these tests assuming that the rogue device is either an AP or a router, the ideas can be applied with minimal modifications to detect rogue ad-hoc networks as well. We omit the description of these modifications and associated details.

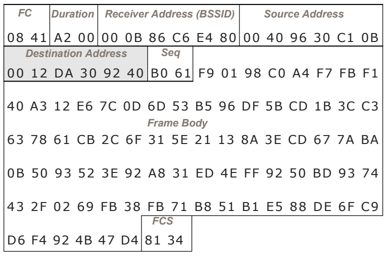
Association Test

To determine whether the suspect is connected to the corporate network, the inference engine directs one of the AirMonitors to attempt to associate with it. If the association is successful, the AirMonitor then attempts to communicate with (e.g., ping) one or more well-known entities that are only accessible from within the corporate network. If this test succeeds, then we know the suspect is attached to the wired network. If the attempt to associate or the ping fails, perhaps because the AP has MAC address filtering or WEP enabled, then we must run more tests.



```
FC     Duration  Receiver Address (BSSID)  Source Address
08 01  D5 00     00 11 95 DA 19 8B         00 08 02 F6 88 AF
Destination Address  Seq    802.2 LLC Header
00 11 95 DA 19 8B    30 19  AA AA 03 00 00 00 08 00
IP Version Hdr Length, Total Length, TTL, ...  Source IP Addr
45 00 00 3C 17 69 00 00 80 01 A1 9F            C0 A8 00 36
Dest. IP Addr
C0 A8 00 32  08 00 6F 33 03 00 DB 28 61 62 63 64
                     Payload
65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74
                                          FCS
75 76 77 61 62 63 64 65 66 67 68 69 16 33 CF DF
```

Figure 3: The highlighted bytes in the IP header of an unencrypted packet reveal the destination address of the packet. In this case it is 192.168.0.50, which was part of our test network.



```
FC     Duration  Receiver Address (BSSID)  Source Address
08 41  A2 00     00 0B 86 C6 E4 80         00 40 96 30 C1 0B
Destination Address  Seq
00 12 DA 30 92 40    B0 61  F9 01 98 C0 A4 F7 FB F1

40 A3 12 E6 7C 0D 6D 53 B5 96 DF 5B CD 1B 3C C3
                 Frame Body
63 78 61 CB 2C 6F 31 5E 21 13 8A 3E CD 67 7A BA

0B 50 93 52 3E 92 A8 31 ED 4E FF 92 50 BD 93 74

43 2F 02 69 FB 38 FB 71 B8 51 B1 E5 88 DE 6F C9
                  FCS
D6 F4 92 4B 47 D4 81 34
```

Figure 4: A WEP encrypted data frame sent to a layer 2 AP. The highlighted bytes are the MAC address of the subnet router on the wired network.

The question of which AirMonitor(s) should be tasked to carry out the association test is a matter of policy. In our current implementation, one or more of the AirMonitors that saw beacons or data packets transmitted from the suspect are selected manually from a central console. We plan to implement an automatic selection policy that will take into account factors such as signal strength of observed packets.

Source/Destination Address Test

This test is used when an AirMonitor can hear data frames that are either destined to or transmitted from the suspect. The inference engine scrutinizes the data frames captured by the AirMonitors for source and destination addresses. If data frames sent to the suspect carry a destination address of a device known to be on the corporate network (or conversely, if frames from the suspect carry such a source address), then we can reasonably conclude that the suspect is acting as an illicit gateway. If the frames are not encrypted, then both the MAC and the IP addresses are available for inspection. If the frames are encrypted, only the MAC addresses are visible.

If we can view the IP addresses, we look at the source/destination IP address of the host with which the device associated with the suspect is communicating. If the corporate network is not using private addresses, we compare the IP address with the known IP subnet ranges on the corporate network to determine if the communication is with a corporate host. See the example in Figure 3.

In any case, we can look at the source or destination MAC address of the packets and compare them with the MAC addresses of devices known to be on the corporate network. If a device associated with the suspect is communicating off the subnet to which the suspect is connected, then the destination (or source, depending on direction of communication) MAC address in their packets will be the MAC address of the subnet router. See example in Figure 4. Otherwise, the MAC address will be from a device directly connected to the subnet to which the suspect is connected.

This test requires a database of the MAC addresses of subnet routers and other devices on the corporate network. To generate this database, the LandMonitors use their routing tables to determine the IP addresses of all routers on the local subnet, and then issue ARP requests to the routers to determine their MAC addresses. The LandMonitors collect the MAC addresses of other devices on the local subnet by continuously listening for ARP requests which are broadcast on the wired network. While switched Ethernet prevents us from easily observing arbitrary traffic on the wired network, we can still observe broadcast traffic from anywhere on the subnet. The LandMonitor periodically summarizes the list of MAC addresses that issued ARP requests, and submits those summaries to the central data collection server.

The MAC address test works only if the suspect is a link-layer forwarder. If the suspect is an IP-layer forwarder, such as a wireless router that combines AP and NAT functionality, the destination MAC address of the wireless traffic will simply be a MAC address belonging to the wireless router. To handle the case of wireless routers forwarding encrypted traffic, we rely on two additional tests described next.

Replay Test

For this test, the inference engine asks one or more of the AirMonitors to play back some of the data frames it overheard with the suspect BSSID. We limit ourselves to playing back data frames that are destined to the suspect device (i.e., the “TO DS” flag in the 802.11 header is true). When the replay test is conducted, the selected AirMonitor replays all the received data frames for either a fixed duration or until a certain threshold has been reached for the total number of unique replayed data frames. To replay an individual data frame, the AirMonitor transmits the entire frame 5 times, and the content is identical each time.

On the wired side, we deploy at least one LandMonitor on each subnet, in such a way that we can sniff all the frames that are headed to the subnet router. The AirMonitor that is about to replay the frame alerts all the LandMonitors before it starts to replay the frames. The LandMonitors start checking to see if multiple identical instances of the same frame appear on the wired network. If multiple instances of a frame are spotted on the wired network during the time when the AirMonitor is replaying frames, the LandMonitor can reasonably conclude that the suspect device is connected to the wired network. We include several heuristics to make sure that spurious retransmissions and certain other types of network traffic do not trigger false alarms.

The replay test is immune to MAC address filtering by the suspect device. It will even work in the presence of encryption, as long as the encryption protocol does not include protections against frame replay. Many wireless security protocols rely on WEP at their core, and are therefore susceptible to replay attacks at least for short durations. We have verified that our replay test works even on Microsoft’s corporate wireless network, which employs 802.1x, as well as IPSec.

As with the association test, we currently manually select the AirMonitors that carry out the replay test.

DHCP Signature Test

The DHCP signature test operates independently of all other tests, and does not include any wireless component. A wireless router device that wants to communicate with other devices on the wired network is likely to issue a DHCP request shortly after it is plugged into the wired network. We use a DHCP LandMonitor that listens to broadcasts of DHCP requests on the wired network, and inserts the summaries of these requests into the database.

The inference engine detects the type of device that issues the DHCP request by parsing the contents of the DHCP requests. DHCP requests can contain a variety of options, and the DHCP protocol allows for the content of some of these options to be highly variable between implementations. Our studies indicate that the contents and ordering of the DHCP options, in particular the parameter request list, can be used as a fingerprint to determine the type and the manufacturer of the device that issued the request. For example, we can distinguish between requests that come from Windows clients, and those that come from wireless routers. In many cases, we can also determine the manufacturer of the wireless router (e.g., DLink, NetGear) If the inference engine detects a DHCP request whose fingerprint does not match any of the device types that are usually connected to the corporate network, or it detects a device whose type and manufacturer does match that of an authorized AP but whose specific MAC address is not on the list of authorized APs of that type, then it raises an alarm.

4.1.2 Guarding Against False Negatives

A malicious user may configure a rogue AP (or a rogue router) to advertise the same SSID and BSSID as one of the authorized AP devices. It becomes much more difficult to detect such devices. Unlike the “false positive” problem that we described in the previous section, this situation presents risk that the DAIR system would not raise an alarm when it really should. We call this the “false negative” problem. To guard against false negatives, the DAIR inference engine can use a number of different techniques to detect a rogue AP that is masquerading as an authorized AP.

First, the inference engine uses historical information to detect anomalies: for example, if a set of AirMonitors suddenly starts hearing an “authorized” AP that they have never been able to hear in past, an alarm is raised. This technique works only when the rogue AP and the corresponding authorized AP are at noticeably different locations.

If the rogue AP and the authorized AP are near each other, we can exploit the sequence number carried in the header of all 802.11 frames to detect the fact that multiple APs are advertising the same SSID and BSSID. An 802.11 sender increments the frame sequence number each time it transmits a frame [28] (excluding retransmitted frames), so successive frames from a sender have monotonically increasing sequence numbers (except when the sequence numbers wrap around). If a rogue AP is masquerading as an authorized AP, the frames sent by the two will intermingle, and therefore successive frames will no longer have monotonically increasing sequence numbers. The inference engine raises an alarm when it detects this occurrence. The sequence number test will work as long as both the authorized and the rogue AP are active simultaneously.

If the attacker manages to disable an authorized AP, and deploys a rogue AP advertising the same SSID and BSSID near the location of the authorized AP, then neither of the above two techniques will detect it. In this case, we can continuously estimate the location of each authorized AP, using the signal strength measurements taken by multiple AirMonitors. If the inference engine notices a significant change in the location of an authorized AP, it raises an alarm. The usefulness of this approach is limited by the accuracy of the location determination algorithm.

4.2 DoS Attacks

Several flaws in the 802.11 architecture can be exploited to generate a variety of DoS attacks on corporate Wi-Fi networks [19, 34]. The DAIR framework can be used to detect a variety of these attacks. We have currently implemented mechanisms to detect two types of DoS attacks.

4.2.1 Deauthentication / Disassociation Attacks

This attack is described in detail in Bellardo et al. [19]. The attacker sends spoofed deauthentication or disassociation frames to either a mobile client, an access point, or both. This will force the victim or victims to exit the authenticated/associated state. Since most wireless drivers on client devices automatically try to re-associate if disassociation occurs, the attacker must continuously generate such spoofed frames to cause significant service disruption. The attack is particularly worrisome, since the attacker only needs to overhear frames from the corporate Wi-Fi network to carry out the attack – it does not need to do any sophisticated decryption.

We have implemented a filter that allows each AirMonitor to submit disassociation frames to the database. The inference engine can easily detect the increased level of disassociation and/or deauthentication frames (perhaps by correlating frames observed at different AirMonitors) and raise an alarm. Furthermore, the inference engine can also provide a rough estimation of the location of the attacker by correlating the signal strength of the disassociation/deauthentication frames seen by different AirMonitors.

4.2.2 NAV Attacks

In Raya et al. [34], the authors detailed several DoS attacks on Wi-Fi networks. In one of the attacks, the attacker continuously sends frames with artificially large duration values in the 802.11 header [27]. The duration field is used to update the network allocation vector (NAV) for any device that hears these frames. Therefore, these large NAV values will force the other transmitters in range of the attacker to withhold their transmissions for extended periods of time.

We have implemented a filter that allows the AirMonitors to submit information about frames with abnormally large duration values to the database. The inference engine further analyzes these frames to raise an alarm if necessary. This mechanism works by measuring the actual duration of the data transmissions and comparing them with the duration values contained in the 802.11 header. This technique is essentially identical to that described in Raya et al. [34]. By correlating observations made by different AirMonitors, the inference engine can also provide a rough estimation of the location of the attacker.

4.3 Summary

We show that the seemingly simple problem of detecting rogue APs is, in fact, quite challenging. We also describe how the DAIR architecture leverages the unique attributes of the desktop infrastructure to limit the number of false negatives and false positive alarms. Our techniques are not foolproof, and we do not guarantee that a suspect is *not* connected to the corporate network. However, we do provide the network administrator with more information, without many false positives and false negatives. We also describe how the DAIR architecture can be used to detect certain types of DoS attacks. We have currently implemented all the mechanisms described in this section, except those used for guarding against “false negatives”. In particular, the ability to calculate an actual location from a list of (AirMonitor, signal strength) pairs is not yet supported by the DAIR system. As such, the location support described in Sections 4.2.1 and 4.2.2 is implemented simply by presenting a list of the nearest AirMonitors ordered by signal strength.

5. EXPERIMENTAL RESULTS

We built the DAIR system, and it is currently operational in several offices on our floor. We begin by providing a detailed description of the hardware and software that we use. Then, we present results that support our argument that dense deployment is necessary for detecting intrusion attacks. Having argued for dense deployment, we then present results that show that the DAIR architecture is capable of handling the dense deployment. Finally, we present results that illustrate certain aspects of the detection techniques described in Section 4.

5.1 Test Environment

Our experiments were conducted on one floor of a fairly typical office building. Our building has rooms with floor-to-ceiling walls and solid wood doors. There is a corporate wireless LAN with six 802.11 a/b/g access points operating on our floor. Our current DAIR system deployment consists of 22 AirMonitors and 1 database server. See Figure 5.

Our database server is Microsoft SQL Server 2005 running on Microsoft Windows Server 2003 SP1. The server hardware is a Compaq Evo D500 with a 1.7 GHz Intel Pentium 4 and 1GB of RAM. The database is stored on a 40 GB 7200 RPM Seagate IDE drive. The drive is formatted as NTFS.

Our AirMonitors run on one of two different types of hardware. One type of machine is a HP Compaq Business Notebook nc6000 with a 2 GHz Intel Pentium M and 512 MB of RAM. These machines run Microsoft Windows XP SP2. We have 6 AirMonitors of this type, located near the six corporate network access points in our building (offices 17, 26, 27, 28, 29 and 30), and used for the deployment density experiments. The other type is a Compaq Evo D500 SFF, identical to the hardware used for our database server. These machines run Microsoft Windows Server 2003 SP1. We have 16 AirMonitors of this type, located in offices numbered 23 and lower, and used for all the other experiments. All AirMonitor machines were equipped with a Netgear WG111U USB 2.0 dongle. This is an 802.11 a/b/g radio with an Atheros chipset.

5.2 Sensor Deployment Density

We argue above that the availability of inexpensive, USB-based wireless cards makes dense deployment of wireless sensors possible. However, we have not addressed the question of what deployment density might be sufficient. The results in this section provide some guidance on this point. Given the seriousness of the rogue AP problem, we would like to maximize the probability of detection, while minimizing the number of false positives and false negatives.

Let us consider a scenario in which an unsuspecting user brings an AP or a wireless router from home, and plugs it into the corporate network. The user uses the AP for his/her own work. In this scenario, both the AP and the user’s wireless NIC are likely to be operating at full power. In other words, the user is not making any attempts to hide from the monitoring system. We would like to know what density of AirMonitors is required to detect such a rogue device.

We first consider a baseline deployment consisting of one AirMonitor placed near each corporate AP on our floor. There are six corporate APs on our floor. Their locations are shown by dark circles in Figure 5. Each AirMonitor listens on 802.11a channel 52.

This deployment provides a good baseline case for two reasons. First, certain commercial products such as Aruba [4] advocate co-locating sensor devices with wireless APs. Second, our corporate wireless network is well-provisioned for both 802.11a and 802.11b/g access, and we can connect to at least one corporate AP from each office on our floor in both 802.11a and 802.11b/g mode.

We set up a rogue AP in 15 offices around the periphery of our building, as well as in two internal conference rooms. The offices are roughly uniformly distributed around the periphery within limits imposed by practical factors such as occupant consent. These offices are shaded in grey in Figure 5.

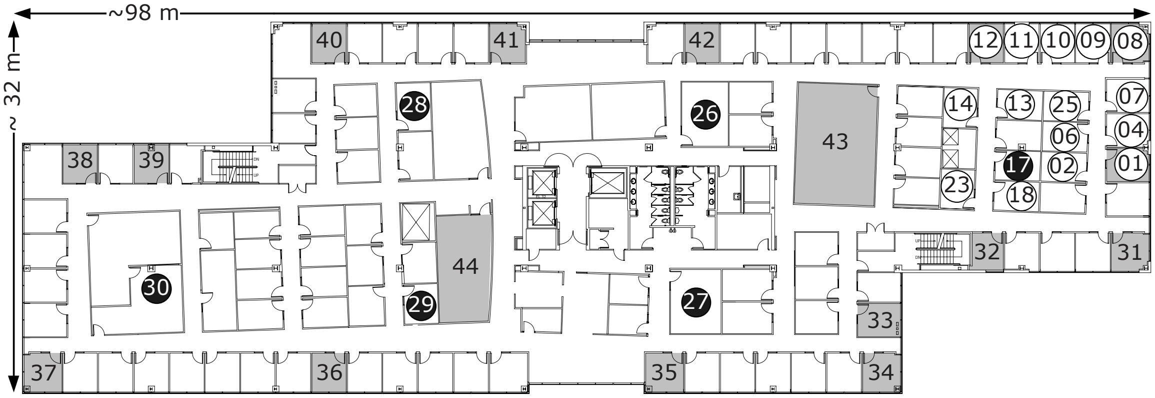


Figure 5: Building map. For the sparse deployment test, AirMonitors were placed in the offices denoted by dark circles; these circles also represent the locations of the six legitimate corporate APs. The shaded offices denote the locations where we temporarily placed a rogue AP. Light circles (and number 17) mark offices containing a denser deployment of AirMonitors used for various other tests.

Rogue AP Location		AirMonitor Location					
		26	27	28	29	17	30
8	Beacons	24	0	0	0	97	0
	Data To AP	0	0	0	0	1.7	0
	Data From AP	0	0	0	0	0.8	0
34	Beacons	0.8	98	0	0	97	0
	Data To AP	0	0.2	0	0	0	0
	Data From AP	0	31	0	0	0	0
37	Beacons	0	0	0	98	0	98
	Data To AP	0	0	0	0	0	0.2
	Data From AP	0	0	0	0	0	76
40	Beacons	7	0	98	98	0	98
	Data To AP	0	0	48	0	0	0.2
	Data From AP	0	0	98	3	0.8	0

Table 1: Percentage of frames overheard by AirMonitors over 60 seconds. 802.11a, channel 52, full power.

In each office, the rogue AP was set to operate on 802.11a channel 52. The AP was broadcasting beacons every 100ms. We also set up a laptop in the same office, that sent and received traffic through the AP. The laptop sent a 1000 byte UDP packet to a wired host every 100ms, and the wired host sent a 1000 byte UDP packet to the laptop every 100ms as well. Thus, 3 frames were sent on this rogue wireless network every 100ms: the beacon, the 1000 byte packet sent by the AP, and the 1000 byte packet sent to the AP. For each office, we check how many frames each AirMonitor overheard within a 60 second interval.

The results for four corner offices are shown in Table 1. We see that for each office, there is at least one AirMonitor that hears most of the beacons. This is true for all offices that we tested, not just the four corner offices. Thus, on our office floor, the baseline deployment of 6 AirMonitors is sufficient to detect the presence of a rogue AP by simply overhearing the beacons.

However, as we discussed in Section 4, to determine whether the AP is connected to the corporate network, we need to look at data frames being sent to or from the AP. The data in Table 1 shows that the AirMonitors were far less likely to overhear data frames. This is not surprising: the beacons were small frames sent at the lowest data rate, while the data frames were much larger, and were sent at higher data rates. Thus, distant AirMonitors were less likely to successfully overhear the data frames. In Figure 6, we show the number of AirMonitors that received a majority of the data frames sent to and from the rogue AP in each office. We see that in at least 4 offices no AirMonitor saw a majority of the data frames sent either to or from the rogue AP.

In the scenario that we just described, the channel on which the rogue AP operated was known in advance, and all the AirMonitors were set on that channel. In reality, each AirMonitor will have to monitor multiple wireless channels, using some scanning schedule. The number of wireless channels that must be monitored is quite large: for North America, the 802.11 standard defines 11 channels in the 2.4GHz band (802.11b/g), and 13 channels in the 5GHz band (802.11a). Even after taking advantage of channel overlap, and spatial overlap in the overhearing range of each AirMonitor, a particular AirMonitor will be on a given channel only for a short amount of time. This is not a problem for detecting beacon frames; for each office, at least one AirMonitor overhears almost all the beacons sent by the rogue AP. However, the probability that an AirMonitor will overhear a data frame sent to or from the rogue AP is much lower. If this probability varies significantly over time, then scanning will further reduce the chance of data frames being overheard.

To illustrate the temporal variability of the probability that an AirMonitor will overhear frames from a particular office, we carried out the following experiment. We set up the rogue AP in office 08, and left it there for a period of 5 hours between 3pm and 8pm on a regular workday. During this time period, the occupant of the office conducted his normal business. The AP was operating on channel 52, as before. We also set up data traffic as described above. We measured the fraction of data frames received by the six AirMonitors for 50 one-minute intervals spread uniformly over these 5 hours. Only the AirMonitor in office 17 overheard any data frames, and the fraction heard in each one-minute interval varied substantially over time, as illustrated in Figure 7. We carried out similar experiments for 802.11g, and the variability results were similar to those for 802.11a.

We conclude from these results that the baseline deployment of six AirMonitors is sufficient for simply detecting the presence of a rogue wireless device on our floor, at least when the device is making no attempts to hide (i.e., transmitting beacons at full power), and the channel on which it is operating is known in advance. However, even in this simple scenario, the baseline deployment of six AirMonitors is not sufficient to guarantee that any data frames sent to or from the rogue AP will be detected. Thus, this baseline deployment is not sufficient to eliminate false alarms using the various techniques that we discussed earlier in the paper. Given the seriousness of the rogue AP problem, we would like to eliminate as many false alarms as possible.

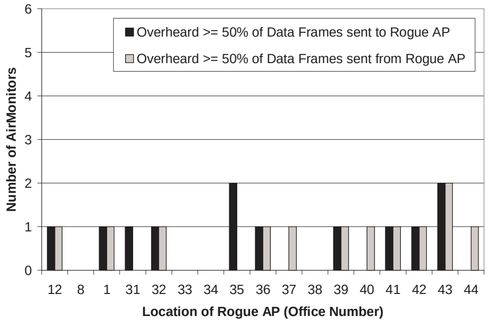


Figure 6: Number of AirMonitors that overheard a majority of the data frames sent to or from a rogue AP. 802.11a, channel 52, full power.

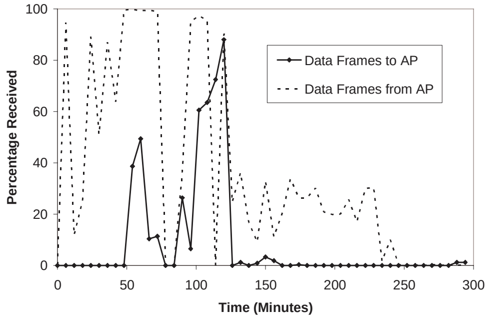


Figure 7: Percentage of data frames sent to or from a rogue AP in office 8, overheard by the AirMonitor in office 17. 802.11a, channel 52, full power.

Next, we consider a dense deployment of AirMonitors. We deployed an AirMonitor in each of 16 offices in one corner of our building. These offices are indicated by light circles in Figure 5, along with office 17. We were unable to place an AirMonitor in the office between 13 and 17, due to logistical issues. Each AirMonitor was tuned to channel 52 in 802.11a band.

We placed a rogue AP in each of these 16 offices that had an AirMonitor. The rogue AP was set to operate at full power on channel 52 in 802.11a band, and to transmit beacons every 100ms. We also set up data traffic as before, so a data frame was sent both to and from the rogue AP every 100ms. We then counted the number of beacons and data frames overheard by each AirMonitor. We found that all the AirMonitors received almost all of the beacons from each office. We also found that a large number of AirMonitors received a majority of the data frames sent to and from the rogue AP in each office. These numbers are shown in Figure 8. We also repeated the experiment for 802.11g, and obtained similar results.

These results are not surprising. It is obvious that the number of AirMonitors that can overhear frames from a particular location can only increase with the density of deployment. Yet, the contrast between Figure 6 and Figure 8 is important. It illustrates the point that with a dense deployment, the number of AirMonitors that can overhear data frames from a particular location increases significantly. Thus, even when AirMonitors are scanning the channels, one can more easily come up with a scanning assignment that can ensure that each office is "covered" by at least one AirMonitor at all times.

It is difficult to achieve such a dense deployment of sensors by using only Access Points and mobile clients as some of the previous work [15] has suggested. Using dedicated sensors for such a dense deployment is also expensive. The DAIR architecture enables dense deployment of sensors at low cost.

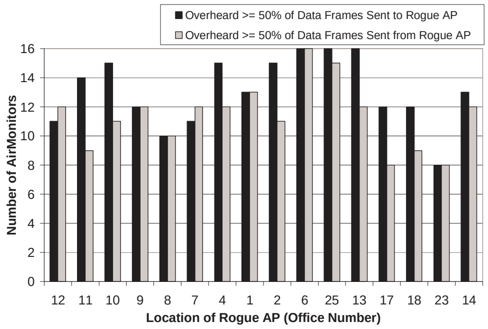


Figure 8: Number of AirMonitors that overheard a majority of the data frames sent to or from a rogue AP. 802.11a, channel 52, full power.

We have also experimented with setting the rogue AP on lower power, and using directional antennas to focus the power away from the AirMonitors. We discovered that even with the baseline deployment of 6 AirMonitors, at least one AirMonitor was always able to hear most of the beacons from the 15 peripheral offices. However, the data frame reception was worse than what is shown in Figure 6. This result further underscores the need for dense deployment of the AirMonitors.

The dense deployment of AirMonitors provides another advantage: the location of the rogue AP or the DoS attacker can be pinpointed with much better accuracy. The accuracy of location estimation provided by systems such as RADAR [17] and Sextant [25] increases with the number of observation points.

5.3 System Scalability

In the previous section, we argued that a dense deployment of AirMonitors is necessary for better system performance. However, a dense deployment brings scalability challenges. In this section, we evaluate the scalability of our system.

We set up 16 AirMonitors in the offices indicated by the light circles in Figure 5. Each AirMonitor was set to listen on channel 1 in the 802.11b/g band. We selected channel 1 because it is the busiest channel in this corner of the building. We ran the 16 AirMonitors for a period of 24 hours spanning a typical workday. Each AirMonitor was running 4 filters: the BSSID filter, the Data Packet filter, the Disassociation filter and the NAV filter. Each filter inserted data into the database at regular intervals. The update interval for each filter was chosen at random between 60 and 120 seconds. Thus, the average update interval was 90 seconds. In addition to being an AirMonitor, the machine in office 01 was also acting as a LandMonitor, and ran a DHCP filter.

In addition to the AirMonitors, one instance of the inference engine was also running. The inference engine issued several queries to the database every 60 seconds, checking for rogue networks as well as disassociation and NAV attacks.

During these 24 hours, the average number of packets processed by an AirMonitor was over 4.9 million. The filters summarized the data quite effectively: we estimate that each AirMonitor, on average, generated less than 2.5Kbps of traffic on the wired network.

The 16 AirMonitors together inserted over 2.5 million rows in the database tables over the 24 hour period. One reason for the large number of rows is that each packet may be overheard and processed by multiple AirMonitors. Because the AirMonitors are submitting summaries of what they have heard, our inference engines use the time of observation to correlate when multiple AirMonitors have

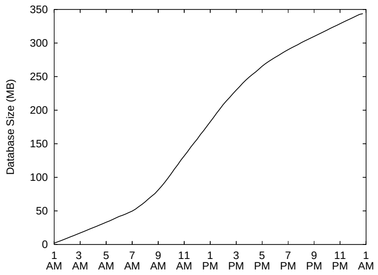


Figure 9: Database size growth over 24 hours.

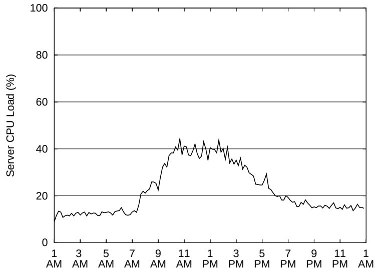


Figure 10: CPU load on the database server, averaged over 10 minute intervals.

heard the same "interesting" event. The growth in the overall size of the database is shown in Figure 9. The database grows by approximately 350MB during this period. As one might expect, the database grows faster during normal working hours. The CPU load on the database server, averaged over 10-minute interval is shown in Figure 10. The peak load is only 43%. Although 350MB may seem large, this data does not need to be kept forever. Future versions of the DAIR system will include a data archiving component that further summarizes the filter data to allow historical analysis, and then discards the filter data periodically.

The latency experienced by insert operations (performed by the AirMonitors) and by query operations (performed by the inference engine) are shown in Table 2. The table illustrates that while a small number of insert and query operations can take several seconds, most database operations take a short time. This result is yet another indication that the database can easily handle the load imposed by our system.

We also experimented with shorter update intervals. When we reduced the update interval by a factor of 6, i.e., each AirMonitor was inserting data in the database every 15 seconds, the load on the database increased by only 10% on average.

In light of these results, we believe that we can easily scale our system by at least one order of magnitude by simply using a faster machine for the database server. There are approximately 100 offices per floor in our building. So one database server will be sufficient to handle data from one floor of our building. Given the nature of wireless networks, the number of queries that the inference engine must perform across multiple database servers (i.e., correlate data across multiple floors or multiple buildings) will be limited. Thus, the system can be scaled further by adding additional servers for other floors.

The growth in size of the database can be handled by archiving data that is more than a few days old to backup devices. The network traffic generated by our AirMonitors is small, and is unlikely to overwhelm the network, or create a network bottleneck near the database server.

	Insert	Query
Median	0.13	0.01
90th Percentile	1.94	0.03
Max	26.96	6.59

Table 2: Latency of database operations in seconds.

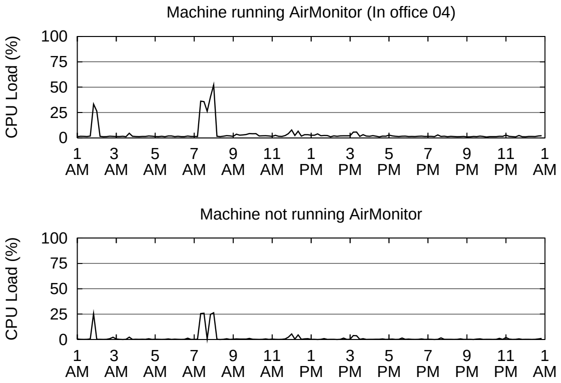


Figure 11: Comparison of CPU load on two machines. The load is averaged over 10 minute intervals.

We now quantify the load imposed by the AirMonitor service on the host machine. We set up a "baseline" machine, in addition to the the 16 AirMonitors. This machine had identical hardware and software configuration to the AirMonitor machines, except that it was not running the AirMonitor software. In Figure 11, we show the CPU load on the baseline machine, and on one of the AirMonitors, over the 24 hour period. We see that the the AirMonitor service imposes little additional load on the machine. In fact, the average additional load on the AirMonitor machine is just 2.25%. The two spikes in the load, one at 2AM, and another at 8AM, are results of administrative activities such a backups and security scans that our IT department performs on all machines connected to the corporate network. We thus conclude that it is indeed feasible to run the AirMonitor service on each desktop machine, without adversely impacting the user's computing experience.

5.4 Demonstrative Results

In this section we present results that illustrate specific aspects of some of the detection techniques discussed in Section 4.

5.4.1 Delay Incurred by the Association Test

The association test identifies APs that do not use security measures to form a wireless network. APs set up by careless employees usually fall in this category. We studied the overhead of this test by analyzing the time taken to return success and failure. We used two APs for this experiment. We disabled all security hooks on the first AP, and enabled WEP with Open Authentication on the second one. Figure 12 shows the steps involved in the association test, and the time taken by each step when connecting to both the APs.

The association test starts by resetting the AirMonitor. It disables promiscuous and monitor mode on the AirMonitor, to allow it to join a wireless network. The AirMonitor then confirms the existence of the rogue network by listening for beacons from the BSSID. If the BSSID is heard, the AirMonitor disables the Win-

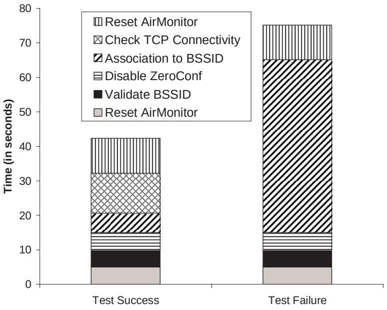


Figure 12: Breakdown of steps in the association test.

Wireless Replays	Replays in Trace	Num Packets	Analyze Trace (s)	Detect 5 Replays (s)
24	20	15270	12.86	0.03
62	44	20071	23.30	4.13
8	6	19932	23.58	4.95
24	18	45956	128.71	2.75
69	47	25273	37.74	1.63
35	28	40975	102.19	1.59

Table 3: Results of the replay test on a router that serves more than 150 APs. Each trace is 30 second long.

dows Wireless Zero Configuration service. (This service associates the wireless card to its own list of networks, and has to be stopped for the duration of the test.) The AirMonitor then sets its channel to that of the rogue BSSID and tries to connect to the rogue network. If association is successful, and the AirMonitor gets a valid IP address, the AirMonitor checks for TCP connectivity to a corporate resource. In our experiments we checked for connectivity to port 80 on the Web Proxy server, which resides on the wired network. The association test concludes by re-establishing promiscuous and monitor mode on the AirMonitor.

The time taken by each of the above steps is illustrated in Figure 12. Note that to ensure successful completion of each step we have provided variable wait periods. This is one of the reasons for the 50 second delay in reporting failure when connecting to an AP. The AP may be far from the AirMonitor, or the wireless medium may be lossy, and so the AirMonitor repeatedly tries to associate to the AP. Despite all these steps, the association test completes in a short period of time. It identifies the unsecure rogue equipment within 45 seconds, and returns failure within 75 seconds. An association failure is a trigger to run the other tests described in Section 4.1.1 to identify the rogue device.

5.4.2 Effectiveness of the Replay Test

We studied the effectiveness of our replay test by replaying packets on one of the APs in our corporate wireless network. We set up an AirMonitor near the AP to replay these packets. Each test lasted 20 seconds and was done in the peak hours of a weekday. We configured the wired router of this AP to capture packets for 30 minutes. This included a 10 minute duration when we did not run the replay test. We performed the replay test 6 times in the remaining 20 minutes. The router serves APs in 9 buildings of our corporate campus, which corresponds to more than 150 APs. Therefore, it sees significant load as shown in the 30 second traces of Table 3.

We analyzed the packet trace at the router and found that there were no replayed packets in the first 10 minutes of the trace. In the remaining 20 minutes, the only replayed packets were in the duration when we were running the replay test. This confirms the effectiveness of the replay test.

We then extracted 30 seconds of the trace for the duration when each of the replay tests were performed. This included a 5 second time before and after the replay test. Table 3 provides more details of our experimental results. A significant number of the replayed packets are captured at the router. Not all reach the router because some of the replayed packets are sent to destinations within the subnet. Further, the size of the trace file captured at the router varied from around 15 to 45 thousand packets, showing a high variation in the activity seen at the router. The time to parse the trace file for detecting replays is proportional to the number of packets in the file and the file size. In this case it varies from 12 seconds to a little over 2 minutes. This overhead is acceptable because the replay test does not need to run continuously; it is executed on-demand, and only when the other tests described in Section 4 fail.

To further reduce the overhead of the replay test, we note that the inference engine need not analyze the entire trace file. Because there are no repeated packets at the router during normal operation, the replay test inference engine can raise an alert as soon as it sees a certain threshold of replayed packets in the trace, and it can then stop analyzing the trace file. Table 3 presents the results of this optimization using a detection threshold of 5 unique packets that are replayed 5 times. As we can see, this optimization significantly reduces the time taken by the replay test to complete.

5.4.3 Effectiveness of the DHCP Test

We have tested our DHCP fingerprinting methodology on the DHCP messages sent by a number of common operating systems and embedded devices, including Microsoft Windows, Apple OS X, FreeBSD, Linux, and home routers from NetGear and D-Link. We found we could uniquely identify the OS or type of the sending device just from the contents of the parameter request list option. For example, current versions of Windows request the DHCP server to send it parameters 1, 15, 3, 6, 44, 46, 47, 31, 33, and 43, while OS X requests parameters 1, 3, 6, 15, 112, 113, 78, 79, 95, and 252. Even if two implementations were to request the same parameters, they are free to list the parameters in any order, and thus are still likely to differ. And while we believe that the parameter list is likely to be sufficient to disambiguate most implementations, some also provide a vendor identifier option in their DHCP messages which identifies them directly (although one of the home routers we tested misidentified itself as “MSFT 98”). Finally, both brands of home routers we tested included a hostname option that contained the model number of the specific device.

To test our DHCP monitoring capability, we ran our monitor on both our own corporate LAN as well as the LAN at a major networking conference. DHCP requests occurred at a leisurely pace on both networks we monitored, the peak load was only a few per minute. The total number of distinct parameter request lists seen over a several day run on our (somewhat homogeneous) corporate network was 22, while the more varied conference network yielded 27 unique requests over several hours of monitoring. Our expectation from this data is that the size of the “known good” signatures dataset which our DHCP monitor must compare against to determine whether to flag an alarm would remain reasonably small.

5.4.4 Threshold for Detecting Disassociation Attacks

The inference engine raises an alarm if one or more AirMonitors sees an abnormally large number of disassociation or deauthentication packets. To determine the threshold for raising an alarm, we conducted some measurements.

First, we monitored wireless traffic at two different institutions at various times of the day, and on various channels. We found that the number of disassociation and deauthentication packets sent every minute never exceeded 5. The median number of such packets seen every minute during the day was 2. Furthermore, no duplicate disassociation packets were seen in any minute.

Next, we wrote a small program to launch a disassociation attack against our corporate network. We monitored the wireless medium for data packets from a specific wireless client, and used the information in the data packets to send a disassociation message to the client and the AP. Using this scheme, we were able to disassociate all wireless clients we tested. However, in the case of most clients, the driver (along with Windows' Wireless Zero Configuration Service) was able to bring the connection back up within a few seconds. Therefore an attacker needs to send disassociation packets at a high rate to cause significant damage to a wireless network.

Given these two measurements, we decided to set the threshold for raising an alarm to 10 packets per minute on any given channel, or 5 packets per minute if all 5 packets are directed to the same client or the same access point. Given these thresholds, there is the possibility both for false positives and false negatives. False negatives are not much of an issue as long as clients reassociate automatically, because if the rate of disassociation messages in an attack is below the threshold, then the client will be getting service much of the time. If our measurements are truly representative of other wireless environments, then we do not expect to see many false positives.

6. DISCUSSION

In this section, we discuss the issues surrounding which channels the AirMonitors should listen on, and we summarize the limitations of our detection techniques.

6.1 Channel Assignment

We argue in the previous section that a dense deployment of DAIR nodes is desirable. However, we did not discuss which channels the DAIR nodes should listen on. This question is complex, and it has been recognized as a key challenge in designing wireless monitoring systems [22].

There are practical limits to how densely one can deploy the AirMonitor nodes. For example, in the scenario that we discussed earlier, we deployed one sensor node per office. Even with such dense deployment, data packets from some of the offices could be overheard in only 8 or 9 other offices. Contrast this with the fact that the 802.11 standard specifies 11 channels in the 2.4GHz band, and 13 channels in the 5 GHz band. Thus, at least some of the AirMonitor nodes must monitor multiple channels. In other words, some sort of scanning schedule must be created.

The scanning schedule should minimize the amount of time any location is left "uncovered" on a particular channel, subject to constraints imposed by the number and the location of the AirMonitors that cover that area. One can take advantage of the fact that sometimes it is possible to overhear data packets on overlapping channels. It may also be possible to use an adaptive strategy in which all AirMonitors rapidly scan all the channels, until a possible problem is detected. At that time, a group of AirMonitors is dedicated to exclusively monitoring the channel on which the anomalous activity is observed. Yet another possibility is to use the proportional time/frame count strategy proposed by Deshpande et al. [22]. We are currently investigating these questions in more detail. Since our current deployment is small, we simply use a static assignment of channels.

6.2 Limitations

The detection techniques proposed in this paper have certain limitations. In this section, we characterize the specific limitations of these techniques. We also discuss the general limitations of the DAIR architecture, as well as the limitations of using the DAIR architecture to build security applications.

In our paper, we have focused on Wi-Fi networks. There are many other kinds of wireless networking technologies in use such as Bluetooth and WiMax. If inexpensive network interfaces for these networking technologies become available, the DAIR architecture could easily be used for monitoring these other kinds of networks as well.

A general limitation of the DAIR architecture, and therefore of the solutions proposed in this paper, is that they will be most effective in an enterprise scenario. In particular, we require that an office building has a moderately dense deployment of stationary desktop computers, each of which has connectivity to the wired network. Furthermore, we require that these desktop computers be under the control of a single administrative entity (or perhaps a small number of administrative entities).

The main drawback of the techniques that we proposed to detect rogue wireless devices is that we can never guarantee a suspect device is harmless. If one of the tests succeeds, then we can conclude that the device *is* connected to the corporate network. However, if all the tests fail, we still cannot say with absolute certainty that the suspect device is *not* connected to the corporate network. In other words, if the person deploying the rogue device has malicious intent and has some technical sophistication, then each of our sequence of tests can be defeated.

Like all monitoring systems, our monitoring system is at the risk if some component of the monitoring system is compromised. Specifically, if some of the desktop machines that we use as monitors are compromised, the attacker can use them to submit false data. Such false data can force our system to generate a large number of alarms. If a large number of AirMonitors are compromised, the attacker can launch a denial of service attack against the database server.

It might also seem that by adding a wireless interface to a desktop computer, we have made the computer more vulnerable, since this new interface may provide an attacker another avenue through which to compromise the machine. However, this is not the case. We have modified the device driver for the wireless interface so that it will not connect to any network except under limited circumstances (e.g., when we ask it to join a specific network during the association test). Even while carrying out such tests, we can closely monitor and control the flow of traffic on the wireless interface. It is, of course, possible that the attacker may compromise the machine by some other means, and replace our driver with another driver that does not have these protections.

7. RELATED WORK

A slew of products for securing corporate networks are available in the market. Firewalls prevent unauthorized users from gaining access to the network [7, 11]. IDSs detect compromised machines in the network [6, 12]. IPSec secures the communication channel between two authorized machines [31], and is frequently used by VPN software. While these techniques are effective in reducing the number of attacks from outside the corporate network, they do not secure the Wi-Fi network against the attacks described in Section 2. In particular, none of these can detect rogue Wi-Fi devices and DoS attacks on Wi-Fi networks. The use of VPNs and IPSec is often not sufficient, as we discussed earlier. IDS products usually detect compromised machines once the attack is launched, and most have a high false positive rate, which significantly reduces their usefulness from the perspective of a network administrator. In comparison, the DAIR security management system detects and locates rogue Wi-Fi devices and various DoS attacks with few false positives and minimal human intervention.

There are several commercial products in the area of corporate Wi-Fi security [1, 2, 3, 13]. Most use one of two approaches: they either rely on APs or they use dedicated and expensive custom hardware sensors for RF monitoring. The marketing literature of these products contains few technical details.

Some commercial products rely on APs for monitoring wireless networks [3]. Although cost effective, this approach has several limitations. First, a single-radio AP can not easily monitor multiple channels since its primary function requires it to spend most of its time on one specific channel serving associated clients. Second, the APs usually have limited CPU power and memory resources, so polling them (i.e., issuing SNMP queries) too frequently is problematic. Third, an AP only provides a view of one end of the wireless communication, so an AP-based solution can not be used to detect problems such as RF holes or excessive interference that primarily affect the client end of the communication. Finally, as discussed in Section 5.2, monitoring the network from an AP alone does not provide comprehensive coverage. To overcome these limitations, some vendors augment the AP-based monitoring by deploying special sensor nodes throughout the organization [1, 2]. However, such specialized sensors are expensive, and require careful planning for an effective deployment.

We are aware of only one prior research paper on detecting rogue devices [15]. In this, mobile clients and APs monitor the network and detect rogue devices. Cisco's Wireless LAN Solution Engine uses a similar approach [8]. The proposed scheme has a few limitations. First, it is difficult to guarantee complete coverage because the monitoring sensors are mobile. Second, the amount of RF monitoring and reporting depends on the battery of the mobile clients. Third, the algorithm proposed in Adya et al. [15] flags any unknown AP as a rogue device, even if the AP is not plugged into the corporate network. Fourth, the proposed techniques do not detect rogue ad hoc networks, and finally, the previous work does not detect DoS attacks on Wi-Fi networks.

There is some prior research on detecting greedy and malicious behavior in IEEE 802.11 networks. Bellardo et al. [19] present a study of various DoS attacks in IEEE 802.11 networks. They demonstrate the attacks, and present simple schemes to counter them. The solution requires clients to cooperate with each other and in some cases, changes to IEEE 802.11. DAIR, on the other hand, detects these faults and reports them to the network administrator. The detection framework for DAIR is more efficient due to the presence of a larger number of sensors.

DOMINO is an AP based solution for detecting greedy behavior in IEEE 802.11 hotspots [34]. Several other researchers have proposed monitoring and characterization of wireless networks by polling the APs [18, 23, 26, 32]. These systems have the same drawbacks as other AP based solutions discussed earlier.

The benefits of dense sensor deployments were presented in Conner et al. [21]. The focus of that paper was on environmental monitoring applications such as temperature control or locating empty meeting rooms, and not Wi-Fi monitoring.

8. CONCLUSION

We present the DAIR system for monitoring enterprise wireless networks using desktop machines. The DAIR architecture takes advantage of the key attributes of the desktop infrastructure: dense deployment, stationarity, wired connectivity, and spare CPU and disk resources.

The first DAIR application we built monitors the corporate Wi-Fi network for security breaches and Denial of Service attacks. We showed that the seemingly simple problem of rogue AP detection is quite complex. We describe in detail how DAIR reduces both false negative and false positive alarms when alerting network managers of security problems.

We built and deployed the DAIR system on a small scale in our offices. We used this deployment to show that a dense deployment of sensors is needed to effectively deal with Wi-Fi security problems. The performance results from this small deployment show that our system can be scaled to larger deployments.

In the future, we plan to expand this initial deployment to cover our entire office building. We are currently building several additional performance monitoring and network management applications using the DAIR framework, and we are extending the DAIR system to support accurate location determination.

9. ACKNOWLEDGMENTS

We thank our shepherd, David Kotz, the MobiSys anonymous reviewers, and Dave Maltz for their feedback on this paper. We also thank Jakob Eriksson for discussions about the rogue AP problem, and we thank Geoffry Nordlund for his assistance with the replay test experiments.

10. REFERENCES

[1] AirDefense: Wireless LAN Security. http://airdefense.net.

[2] AirTight Networks. http://www.airtightnetworks.net.

[3] AirWave Management Platform. http://airwave.com.

[4] Aruba Wireless Networks. http://www.arubanetworks.com.

[5] BAE Systems, Frequency Selective Surface Panels. http://www.baesystems.com/atctowcester/products.htm.

[6] Cisco Intrusion Prevention System. http://www.cisco.com/en/US/products/sw/secursw/ps2113/.

[7] Cisco PIX 500 Series Security Appliances. http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/index.html.

[8] Cisco Wireless LAN Solution Engine (WLSE). http://www.cisco.com/en/US/products/sw/cscowork/ps3915/.

[9] CMU Warning Against Multiple Active Interfaces. http://www.cmu.edu/computing/documentation/connect_wire_wireless/wired_wireless_rules.html#multiple.

[10] Private communication with Microsoft IT department.

[11] Symantec Enterprise Firewall. http://enterprisesecurity.symantec.com/products/products.cfm?productid=47.

[12] Symantec Network Security 7100 Series. http://enterprisesecurity.symantec.com/products/products.cfm?productid=540.

[13] Symbol Technologies: SpetcrtumSoft Wireless Management System. http://www.symbol.com.

[14] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol. RFC 3748, IETF, June 2004.

[15] A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and Techniques for Diagnosing Faults in IEEE 802.11 Infrastructure Networks. In *Proceedings of the Annual ACM International Conference on Mobile Computing (MobiCom)*, September 2004.

[16] P. Bahl, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill. DAIR: A framework for managing enterprise wireless networks using desktop infrastructure. In

Proceedings of the Annual ACM Workshop on Hot Topics in Networks (HotNets), November 2005.

[17] P. Bahl and V. N. Padmanabhan. RADAR: An in-building rf-based user location and tracking system. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, March 2000.

[18] M. Balazinska and P. Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of the Annual ACM/USENIX International Conference on Mobile Systems, Applications and Services (MobiSys)*, May 2003.

[19] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *Proceedings of the USENIX Security Symposium*, August 2003.

[20] N. Cam-Winget, R. Housley, D. Wagner, and J. Walker. Security flaws in 802.11 data link protocols. *Communications of the ACM*, 46(5):35–39, May 2003.

[21] W. S. Conner, L. Krishnamurthy, and R. Want. Making Everyday Life Easier Using Dense Sensor Networks. In *Proceedings of International Conference on Ubiquitous Computing (UbiComp)*, October 2001.

[22] U. Deshpande, T. Henderson, and D. Kotz. Channel sampling strategies for monitoring wireless networks. In *Proceedings of the Second International Workshop On Wireless Network Measurement (WiNMee)*. IEEE Computer Society Press, April 2006.

[23] Diane Tang and Mary Baker. Analyis of a Local-Area Wireless Network. In *Proceedings of the Annual ACM International Conference on Mobile Computing (MobiCom)*, August 2000.

[24] J. R. Douceur and W. J. Bolosky. Progress-based regulation of low-importance processes. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, December 1999.

[25] S. Guha, R. Murty, and E. G. Sirer. Sextant: A unified node and event localization framework using non-convex constraints. In *Proceedings of the Annual ACM International Conference on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2005.

[26] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of the Annual ACM International Conference on Mobile Computing (MobiCom)*, September 2004.

[27] IEEE80211. *IEEE Standard for Wireless LAN-Medium Access Control and Physical Layer Specification, P802.11*.

[28] IEEE802.11b/D3.0. Wireless LAN Medium Access Control(MAC) and Physical (PHY) Layer Specification: High Speed Physical Layer Extensions in the 2.4 GHz Band.

[29] IEEE802.11i. IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications - Ammendment 6: Medium Access Control (MAC) Security Enhancements, 2003.

[30] IEEE802.1X. IEEE Standard for Local and metropolitan area networks, Port-Based Network Access Control, 2004. http://www.ieee802.org/1/pages/802.1x.html.

[31] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *IETF RFC 2401*, November 1998. http://www.ietf.org/rfc/rfc2401.txt.

[32] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. *Wireless Networks*, 11:115–133, 2005.

[33] M. Milner. NetStumbler WLAN detection software, 2004. http://www.stumbler.net.

[34] M. Raya, J.-P. Hubaux, and I. Aad. DOMINO: A System to Detect Greedy behavior in IEEE 802.11 Hotspots. In *Proceedings of the Annual ACM/USENIX International Conference on Mobile Systems, Applications and Services (MobiSys)*, May 2004.