Proceedings of the 2003 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY June 2003

Wireless Intrusion Detection and Response

Yu-Xi Lim, Tim Schmoyer, *Snr. Member, IEEE*, John Levine, *Member, IEEE*, and Henry L. Owen, *Member, IEEE*
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Ga. 30332-0250

Abstract – A prototype implementation of a wireless intrusion detection and active response system is described. An off the shelf wireless access point was modified by downloading a new Linux operating system with non-standard wireless access point functionality in order to implement a wireless intrusion detection system that has the ability to actively respond to identified threats. An overview of the characteristics and functionality required in a wireless intrusion detection system is presented along with a review and comparison of existing wireless intrusion detection systems and functionalities. Implemented functionality and capabilities of our prototyped system are presented along with conclusions as to what is necessary to implement a more desirable and capable wireless intrusion detection system.

Index terms – Network Security, Wireless Intrusion Detection

I. Introduction

Increasing numbers of organizations are deploying wireless networks, mostly utilizing the IEEE 802.11b protocol. Even though attempts have been made to secure these networks, the technology used is intrinsically insecure and still highly susceptible to active attacks and passive intrusions.

Standard tools for monitoring wired networks and ensuring their security examine only network (layer 3) or higher abstraction layers based on the assumption that the lower layers are protected by the physical security of the wires. However, this assumption cannot be extrapolated to wireless networks because of the broadcast nature of such networks. Ideally, an intrusion detection system for wireless networks should function at the datalink layer (layer 2) or even lower if extremely high security is required. Yet, to go beyond mere detection and to actually provide useful protection for the network, it might be necessary to actively disable unauthorized clients attempting to access the network.

This paper briefly surveys wireless intrusion methods, wireless intrusion detection, and wireless intrusion response and discusses a practical implementation of a wireless intrusion detection system. The focus is on the popular IEEE 802.11b standard but methods discussed may be extended, to varying degrees, for the other IEEE 802.11 standards. The paper does not detail other means of attacking wireless networks which do not exploit weaknesses in the protocol or other inherent weaknesses, for example, by compromising the clients or by trying default passwords on poorly configured equipment.

II. Background

A. Intrusion Methods

Signals from wireless networks are usually omni-directional and emanate beyond the intended coverage area. Such properties make the physical security of the network mostly impractical. Many passive and active intrusion methods quickly arose to abuse this weakness. Passive methods use radio frequency (RF) monitoring and do not broadcast any signals. Active methods may merely broadcast signals to query the status of the network, or they may even insert malicious data into the network to cause disruptions. This is a description of the most common methods and is by no means exhaustive, especially since new exploits and tools appear every week.

The most common wireless intrusion method is "Wardriving". This is usually done using a Windows laptop running Wardriving software, such as NetStumbler, and equipped with an IEEE 802.11b adapter and external antenna. The "Wardriver" drives around high-tech neighborhoods hoping to detect IEEE 802.11b signals that have leaked out onto the street. NetStumbler looks for beacon frames from the access points (APs). From these beacon frames, it is typically possible to determine the encryption strength, channel, and type of hardware used. If the network is unsecured, the Wardriver may also record other details of the network like the Service Set Identifier (SSID). In many cases, this is performed by hobbyists and no further invasive action is taken. Such hobbyists would generally combine the data with Global Positioning System (GPS) information to produce geographic maps of wireless networks in the area and their configurations. There are other less common software available for Wardriving, depending on the platform used. dStumbler runs on BSD systems, MiniStumbler runs on PocketPC handhelds, Kismet runs on several platforms, and Wellenreiter runs on Linux systems. Depending on the software used, Wardriving

ISBN 0-7803-7809-1/03/$17.00

may be passive or active. Active software like NetStumbler, dStumbler, and MiniStumbler actually broadcast probe request frames to elicit responses from APs [1]. This improves their chances of detecting APs, especially when the Wardriver is in a moving vehicle. Passive methods merely perform RF monitoring to detect chance signals from the APs.

Another popular intrusion method concerns the infamous weakness in the Wired Equivalent Privacy (WEP) encryption used by IEEE 802.11b networks [2, 3]. This is usually the second stage of an intrusion following detection of a secured AP by Wardriving. The most commonly used tool for WEP key extraction is the Linux program AirSnort [4]. An intruder using AirSnort would surreptitiously collect wireless network traffic of the target network. When enough frames have been collected from the network, AirSnort can determine the WEP key of the network by examining the "weak" frames. It usually takes only a few hours to collect enough frames. Manufacturers have released updated firmware that addresses the transmission of such weak frames; however, a network remains vulnerable if a client continues to use an outdated wireless network adapter. A less common alternative to AirSnort is WEPCrack [5], but this program has less features and lower accuracy. AirSnort is a passive monitor and does not emit any signals.

On many networks, intrusions are not limited to unauthorized clients but could include unauthorized APs. Often, these "rogue" APs might be installed by valid users attempting to increase the range of the network but doing so without proper authorization. This usually results in a security hole that may be exploited by intruders. A more relevant scenario would have an intruder planting an AP with a higher than normal broadcast power to masquerade as a legitimate AP. Unknowing clients would attempt to associate with this AP believing it is valid. The intruder could then use information collected from these association attempts to determine network security settings and other aspects of the network.

Also possible is a denial-of-service (DoS) attack on the network. This could occur in several ways, the most primitive being the use of radio equipment to broadcast noise at the 2.4 GHz operating frequency of the network. This would cause the network to drop frames, eventually to the point of total collapse. A more refined method would be to broadcast invalid frames to either clients or APs, or even to both. The clients or APs would respond to these invalid frames and, if present in sufficient number, these invalid frames could interrupt the flow of normal traffic.

A few other methods are proof-of-concept and have not been observed frequently in real networks. The first is the man-in-the-middle attack using Address Resolution Protocol (ARP) poisoning [6]. This uses a known vulnerability on Ethernet networks concerning unauthenticated ARP messages. Many systems have been developed for wired networks to counteract such poisoning but administrators often forget to extend this protection to wireless bridges which could also serve as entry points for such attacks.

A different method was demonstrated by 802.11ninja during DefCon in 2001 [7]. Using a program called Monkey Jack, management frames were sent to wireless clients at the convention forcing them to disconnect from valid APs and re-associate instead with a bogus AP managed by the attackers. The attackers also offer code on their website to exploit other vulnerabilities even in wireless Virtual Private Networks (VPNs). All these rely on unauthenticated message vulnerabilities on IEEE 802.11b networks.

B. Existing Systems

There currently exist a few products that perform the intrusion detection and active response roles for the above attacks. However, none provide adequate protection for wireless networks, especially for larger deployments.

AirDefense [8] is a complete hardware and software system consisting of sensors deployed throughout the network, which are interfaced to a management appliance, and adminstered by a management console. Their starter kit provides five sensors and can guard up to ten APs. AirDefense detects intruders and attacks and also diagnoses potential vulnerabilities in the network like misconfigurations. The manufacturer claims that AirDefense can detect most of the threats mentioned above. Also, AirDefense offers other management functions such as fault tracking and inventory auditing. The company is also launching a new product that offers active responses to intrusion attempts and can integrate with the AirDefense product. Their system forces an intruder to dissociate from the valid network and optionally re-associate with a "honey pot" AP. The combined AirDefense and ActiveDefense systems would come closest to our ideal system described later.

Another commercial product is AirMagnet [9] which runs on laptops or handhelds and also includes a Cisco wireless card in the package. Like AirDefense, it incorporates detection of vulnerabilities and intrusions. For intrusions, AirMagnet detects unauthorized APs and clients and DoS attacks by flooding. A similar product is Surveyor Wireless [10]. These software products require a technician to move around the network to detect possible security threats. Interestingly, this software may also be used by an intruder, though such use is unlikely because of the high price.

ISBN 0-7803-7809-1/03/$17.00 © 2003 IEEE

One non-commerical product is Fake AP [11]. Fake AP is a simple Linux program that simulates a user-specified list of APs by broadcasting IEEE 802.11b beacon frames. This potentially confuses an intruder passively sniffing the network. The program is available freely under the GNU Public License (GPL).

AirSnare [12] is a program for Windows that detects DHCP requests or unauthorized MAC addresses attempting to connect to an AP. Intrusion response consists of an alert to the administrator and optional message is sent to the intruder via Windows netmessage. AirSnare has a non-commercial license.

III. Proposed New Architecture and Capabilities

An ideal system combines the functionality from the products described above and also implements some novel features. The proposed system described below is intended for IEEE 802.11b networks but could be extended to other specifications with some modification.

A. Physical Specifications

In the complete system a number of devices would be deployed throughout a wireless network. Each device should be located near existing APs to provide similar coverage. The device would be a single unit with a form-factor similar to a conventional AP. These devices would be connected to a standard wired network to allow for secure remote management. Since these devices are intended to be cheap to deploy and simple to maintain, they would be limited in capabilities and instead rely on a central server on their wired network to perform additional tasks like logging, similar to the AirDefense system.

Each device would ideally be an integrated system running on low-cost, low-power embedded processors and using standard hardware. While the capability to simultaneously monitor all channels used by IEEE 802.11b would be useful, it is not possible using a standard wireless card. Simultaneously monitoring of all channels would require multiple cards or a specialized card with a fast digital signal processor to decode the signals. Practical applications would rarely require multi-channel capability. NetStumbler and similar programs broadcast on all channels, and thus can be detected by monitoring any given channel. Also, most of the interesting traffic occurs on the channel used by the valid network.

B. Intrusion Detection

The primary function of our proposed new device would be intrusion detection. This would happen at different levels. The most basic level would be to track the Media Access Control (MAC) address of network adapters attempting to associate with the network. If the MAC address does not occur in the whitelist or is blacklisted, it is flagged as a possible intruder. Such a procedure is commonly known as MAC filtering and might not be practical in a large organization where users may employ their own wireless cards. Fortunately, MAC addresses are not totally random. The first three bytes are specific to each manufacturer and manufacturers usually utilize only a small range of the available addresses. By checking each MAC address against such patterns, it would be possible to determine forged addresses randomly generated by intruders. It is possible for users or attackers to change MAC addresses reducing the effectiveness of using patterns.

It may also be possible to dectect passive intruders using the IEEE 802.11b Request to Send (RTS) and Clear to Send (CTS) frames. Normally these frames are used to determine if the medium is clear and to reserve a block of time to send the data. RTS is acknowledged with a CTS by firmware and is usually beyond the control of the user's software. The RTS and CTS relationship may be used as a means of detecting intruders that are present on the network. For example, if an active Wardriver is detected, the MAC address could be logged. Subsequently, RTS messages could be sent to that MAC address. If the intruder is now passively collecting data the card may still respond with a CTS, revealing its presence.

Stateful monitoring of traffic could provide clues about intrusions. Unusual data like unsolicited random responses could indicate an intruder probing a network. Such anomaly tracking has already been implemented on higher network layers. This technique may be extended to the IEEE 802.11b protocol on events like authentication and association, or even RTS/CTS layer 2 messages.

It should be possible to determine unique signatures for each kind of attack. Even within a class of active attack, like Wardriving, signatures can be used to identify the specific software used. Such signatures have been described for common Wardriving software and include characteristics from sequence numbers, control types and subtypes, destination MACs, Service Set Identifiers (SSIDs), Organizationally Unique Identifiers (OUID), Logical Link Control (LLC) protocol types, LLC protocol identifiers, and even data payload [1].

To improve detection accuracy, it should be possible to utilize any number of algorithms to profile the attack,

ISBN 0-7803-7809-1/03/$17.00 © 2003 IEEE

including rule-based algorithms, expert systems, or even artificial neural networks that "learn" the normal behavior of the network. To minimize the requirements of each device and to improve detection accuracy by polling more devices, additional intrusion detection logic could take place on the central server where more information and more processing capabilities are available.

If several devices are used and are connected to a central server, it would be possible to triangulate the position of an attacker or rogue access point. The position and even the motion can be taken into account in determining if the source is merely a valid user with an unregistered MAC address or an intruder outside the premises. The central server could also correlated wireless authentication with authentication on other security systems. For instance, authentication and association on the wireless network could be cross-referenced with Remote Authentication Dial-In User Service (RADIUS) authentication to establish if a valid interface card is actually being used by its assigned user and not someone else.

C. Intrusion Response

Standard passive responses are typical of intrusion detection systems. They include logging of the intrusion, real-time notification, or even disabling the entire network. The proposed intrusion detection system would include such passive responses.

However, effective intrusion response on wireless networks goes beyond merely passive responses to the intrusion attempt. Additional deterrents are required because of the reduced physical security on such networks. Of course, it would be still be necessary to log intrusion attempts for analysis, even with effective active countermeasures. Thus, the ideal wireless intrusion detection system would respond actively to threats.

The most effective defense against intruders would use the previously described threats against the attackers themselves. This would work against attackers that have configured their network interfaces to authenticate on the network. Both ARP poisoning and disassociation-reassociation would work well in such cases.

DoS attacks against the intruder by flooding would have an adverse effect on the overall network performance. Even if the intruder is on a different channel from the rest of the network, interference can still occur between channels. Thus flooding DoS attacks are not recommended against an intruder.

A possible alternative to flooding DoS attacks are to utilize specially crafted malformed frames directed specifically at the intruder. These could exploit poorly defined or implemented aspects of the IEEE 802.11b specification which may result in crashing the software on the intruder's computer. If the intruder has authenticated successfully on the network and is actively receiving and transmitting data, it might be possible to attack the intruder using various techniques. By fingerprinting an active intruder during the intrusion detection phase, the operating system and software configuration of the intruder can be determined. This can be supplemented with information obtained via TCP/IP fingerprinting and port scanning. In this way, the intruder can be profiled for known vulnerabilities and these can then be exploited against the intruder. The details of these vulnerabilities are beyond the scope of this paper.

Passively listening cards are placed in monitor mode and transfer virtually all received frames to the software for processing. So a passive intruder could merely log frames without acting on them and the frames would have no effect on the network configuration of the intruder. Techniques that exploit unauthenticated ARP messages or IEEE 802.11b management frames are thus rendered useless when the intruder uses monitor mode and chooses to ignore these messages or frames.

It is still possible to confuse an intruder using decoys. Adopting Fake AP's functionality, the device we have proposed and prototyped will broadcast falsified AP information to an intruder running NetStumbler or similar programs. If the intruder uses WEP key extraction software, the device would broadcast weak frames containing random data, thus confusing the algorithm used in the WEP key extraction. Other means of deceiving passive attackers include broadcasting false management information that may contain specific details like fake IP addresses. If the intruder later attempts to use the falsified information to connect to the network, it would be easier to identify the intrusion. Tools developed to perform such functions on higher network layers could also be used.

An effective intrusion detection system should include many types of decoys. Again, the broadcast of such decoy frames on a shared medium could adversely affect overall network performance so it should be used sparingly and with caution.

IV. Experimental Prototype Implementation

An experimental prototype system was assembled using commercially available parts and freely available software. The basic hardware requirements (low-powered processor and wireless and wired network connectivity) meant that modification of a standard wireless access point would be the easiest route. The USRobotics USR2450 was chosen for this purpose, taking into

ISBN 0-7803-7809-1/03/$17.00

account that it was based on the Eumitcom WL1100SA-N board, which in turn is supported by OpenAP [13]. OpenAP is a Linux distribution for the access point and allows for a high degree of customization. Also, the WL1100SA-N boards utilize an x86-compatible AMD Élan SC400 and a standard PC Card IEEE 802.11b network adapter. Thus, the software developed for this access point could be easily ported to a more powerful platform, like an x86 PC with a PC Card wireless network adapter. The USRobotics USR2450 together with Open AP's Linux, allows us to download non-standard functionality to the off-the-shelf commercial device.

For the prototype, only one unit was assembled and the central management server was omitted. Management was instead done on the unit itself via a web-based interface. It was originally intended that the device retain its AP functionality after modification. It was later determined that, due to hardware and software constraints, some original AP functionality had to be sacrificed for the purposes of being able to easily prototype some of our proposed concepts.

Due to limitations in the processing power and memory available on the unit, the detection of only selected intrusions was implemented. The focus was on NetStumbler detection, since it is the most commonly used tool. For response, the unit was capable of logging intrusions and deploying active counter-measures including decoy frames for AirSnort, fake AP probe response frames, and a DoS attack against NetStumbler.

Another limitation that was encountered was that since a standard IEEE 802.11b card was used, the prototype could not monitor all channels at once. Instead, it had to switch frequently between channels to achieve the desired effect. Also, the firmware restricted the device to functioning in monitor mode instead of Host AP mode as was originally intended.

A. NetStumbler DoS

In the course of developing the module for fake AP beacons and probe responses, a vulnerability was discovered in the IEEE 802.11b implementation. Probe responses contain a variable length list of tagged fields. Each field contains a tag to identify the contents, the length of the field, and the actual data itself. SSIDs in practice may be set to null. However, when actually being sent as a tagged field, a null SSID would be transmitted as a single ASCII space character.

During testing the space character in the SSID was mistakenly omitted and it was observed that this caused the Windows PC running NetStumbler to lose its connection to the wireless card. Depending on the version of Windows used, the computer may even respond sluggishly. Further investigation revealed that this effect was the same even if another program other than NetStumbler was used to scan the network for access points. Furthermore, this vulnerability affected a variety of systems, running both Windows and Linux and using wireless adapters from several manufacturers. The DoS effect we created varied from only requiring the wireless card to be ejected and reinserted to hanging of the software and causing the system to slow down. One setup that was not vulnerable was a Windows XP PC with a card using Intersil's PRISM 3 chipset. PRISM 3 cards were open to attack when used on PocketPC and Linux platforms. Fortunately, Wardriving with a Windows XP laptop requires the use of NetStumbler which consequently requires the use of a wireless card using the ORiNOCO chipset and this setup is susceptible.

As this technique could also affect valid clients attempting to scan and associate with the network, it is only used when NetStumbler has been detected. And even if the intruder has associated with the network and is no longer scanning, it would be possible to send a disassociation frame to the intruder and cause it to resume scanning. At this point, the malformed probe response attack can be used.

Obviously, this vulnerability should be addressed by manufacturers before it is exploited maliciously. We decided to use this technique in our prototype systems to actively prevent wireless intruders.

B. AirSnort Decoys

Examining AirSnort's source code revealed that it uses a simple algorithm to determine if a frame is weak. Weak frames are then added to a pool to be analyzed later by another algorithm to determine the WEP key.

The weak frame detection algorithm relies on simple pattern matching of the initialization vector (IV) of the frame. The IV itself consists of three bytes. AirSnort considers a frame weak if the first byte of the IV has a value between 2 and 16 inclusive and the second byte has a value of 255. Each three-byte pattern is only added once to the pool.

Creating decoys for AirSnort is a simple task. The decoy frames contain mostly valid data except that the IV is a randomly generated number that matches the criteria described above. Also, at least one byte of "encrypted" data should be randomly generated. AirSnort requires only a few such frames and would cease capturing frames when the key has been extracted. Our implementation used sequential numbers for the third byte while the first byte and encrypted data are a single random byte each.

ISBN 0-7803-7809-1/03/$17.00 © 2003 IEEE

Other required fields are also filled in. First, the frame is marked as an encrypted data frame. The BSSID is the actual one configured by the user and the source address is also set from this value. The destination is randomized since it is ignored by AirSnort. To prevent degradation of the network, the decoy frames are only sent intermittently at a user-controlled rate. Given that the system only has to protect the valid network from intrusion, the decoys are broadcasted only on the valid channel.

C. Fake Probe Responses

The functioning of this mechanism is similar to the Fake AP program. Probe responses are broadcasted with bogus access point information. For example, the Service Set Identifier (SSID) and Base Station Set Identifier (BSSID) would be randomized.

Attempting to flood NetStumbler by broadcasting thousands of random responses is unrealistic and often it is not difficult to pick out the real access point which would be the only steady response. Our prototype mimics Fake AP behavior and instead selects from a user-specified list of SSIDs and MAC addresses. The use of predefined MAC addresses as opposed to random ones enhances the illusion since these MAC addresses can be similar to valid ones with valid OUIDs. The SSID and MAC are transmitted on the channel that a NetStumbler probe request is detected. This appears as several networks operating constantly on all channels that NetStumbler scans. Since the probe response is destined only for the intruder, valid clients should have no problems connecting to the network.

D. Detection and Response

The unit must first be configured for the valid network's parameters, including SSID and MAC address, also known as the Base Station Set Identifier (BSSID), of the AP through the web-based management console. The range of responses can be toggled independently through the console too. If AirSnort decoy frames are enabled, the device would begin transmitting the decoys at user-determined intervals.

The device will be screening traffic on all channels and keeping track of probe requests, association messages, and authentication messages. If a computer running NetStumbler is present, NetStumbler's probe requests will be detected on the device. When the frequency of probe requests exceeds a user-defined threshold, the computer is logged as a NetStumbler computer. Intrusion response will be activated if enabled. Intrusion response includes DoS and fake probe responses.

Association and authentication are also monitored to determine valid clients. Each client is identified by its unique MAC address. For each MAC address, a state machine tracks the current association/authentication state. Clients that have fully associated and authenticated with the network are assumed to be valid. If they were previously flagged as NetStumblers, this flag would be cleared and any intrusion response against them would cease. The device would only consider authentication and association frames intended for the valid network so as to minimize false negatives.

E. Results

Tests were conducted with a network consisting of a D-Link DWP900AP access point, the prototype device, a PC with an ORiNOCO Gold wireless card running NetStumbler, and two Linux PCs with Linksys WPC11 wireless cards acting as valid clients.

The device was successful in detecting NetStumbler and did not give any false positives with valid clients using the default threshold of 30 probe requests per 10-second interval. However, when the NetStumbler computer is moving at some speed, as would happen if the intruder were driving past an installation, the detection accuracy diminishes. This is because of the reduced time in which the intruder is present on the network. Fortunately, fast-moving NetStumblers also suffer from reduced accuracy while scanning networks. The detection accuracy was also affected by NetStumbler's scanning speed – the delay between consecutive probe request transmissions. An intrusion was detected when using the ORiNOCO configuration utility to scan the network. Other active Wardriving tools like dStumbler were not tested since they function along similar lines and the system was assumed to be able to detect them with at most slight modifications to the threshold.

The DoS attack against NetStumbler was very successful and would disable the system within a few seconds. The AirSnort decoys also worked as expected. When the device was configured to broadcast decoys every 30 seconds, AirSnort would extract an invalid key after less than an hour. However, the impact of the AirSnort decoys on network performance was to lower throughput by about 1.9%. Fake probe responses were detected by NetStumbler but appeared as intermittently functioning access points, which is undesirable.

F. Discussion

The main limitation in our prototype device is the use of a standard wireless network adapter. This adapter can only examine traffic on one channel at a time and there is a noticeable delay when switching channels. This limitation

ISBN 0-7803-7809-1/03/$17.00 © 2003 IEEE

means that the unit can only detect NetStumbler if by chance NetStumbler is transmitting on the same channel as that which the unit is scanning. Because some probe requests from NetStumbler are not detected, no probe response is sent and thus NetStumbler flags the AP at that channel as disabled. Thus, the bogus access points appear active only intermittently on NetStumbler and this behavior could be used to discriminate the actual access point from the decoys. Also, the device might be scanning another channel while a client is authenticating or associating on the valid channel. This loss of information makes it difficult to accurately track the association and authentication states of each client. Channel hopping on the prototype is user-configurable, since certain administrators would prefer to collect full authentication and association data rather than monitor all channels.

As mentioned earlier, firmware limitations restricted our choice of operating modes for the IEEE 802.11b adapter in our prototype device. In Host AP mode, the device would be able to function as an access point. However, in this mode the card filtered out frames not destined for its MAC address, which was an undesirable "feature". In monitor mode, the card would not be able to respond to low-level frames and handling these frames in software proved too slow to meet timing specifications. In the end, we decided to use monitor mode, forgo AP functionality, and to ignore low-level frames where possible.

In addition, the firmware limitations ruled out the possibility of other DoS attacks against the intruder. One possible DoS would require the use of malformed frames with invalid duration or CRC values. Unfortunately, this could not be completed on the prototype because the firmware would not transmit such frames. Attempts to send frames to non-existent MAC addresses were also foiled because the firmware would require a CTS from the non-existent MAC before sending the frame. With modified firmware, it might be possible to confuse the intruder with such out-of-range values. We did not pursue accessing and modifying the firmware in this version of our prototype.

The impact of AirSnort decoys on network performance is inevitable and is, in our opinion, within tolerable values. On high throughput networks, the number of decoy frames is insignificant compared to the overall number of frames. On networks with less traffic, the transmission frequency of the decoys can be reduced. This would not affect their effectiveness as the decoys only have to outnumber the similarly reduced number of actual weak frames.

V. Conclusions

We downloaded a replacement Linux operating system and modified functionality into an off-the-shelf USRobotics USR2450 wireless access point with the intent of prototyping our ideas. Even though the prototype functioned nearly as intended, it still falls short of a complete wireless intrusion detection and response system.

The limitations of processing power and memory inside our modified USRobotics USR2450 meant that only few forms of intrusions could be monitored. To be effective against the current repertoire of attacks, the processing power of the device needs to be improved. Our prototype used a processor that is equivalent to an Intel 486 processor and had only 4 megabytes of flash memory. Since the device would ideally be low-cost and have low power consumption, a cheap integrated embedded processor is the most likely choice. There are many processors available that provide the necessary IO interfaces and greater processing power. Flash memory and RAM are available at very low costs too, so memory restrictions should be less severe in future devices.

The ability to receive or transmit on only one channel is a more significant obstacle. One possible solution would require the use of at least two cards. One card would perform scanning of all channels while another card would transmit and receive only on the valid channel. Having multiple cards would also alleviate the problem of having no traffic on the simulated networks since extra cards could be used to transmit random or simulated traffic to enhance the appearance of a complete and functioning network.

The prototype was a standalone solution. Integration with other similar units and a central server should pose little problem. The threshold detection algorithm performed satisfactorily under lab conditions, but on larger, ill-defined networks, an adaptive algorithm would be required and this could run on the server. Having multiple devices to perform detection could also improve the reliability of detecting fast-moving Wardrivers.

VI. Acknowledgements

The authors would like to acknowledge that this research was initially completed as part of a design project by Georgia Institute of Technology undergraduate students Seng Oon Toh, Nitin Namjoshi, Varun Kanotra, and Yu-Xi Lim.

VII. REFERENCES

[1] J. Wright, "Layer 2 Analysis of WLAN Discovery Applications for Intrusion Detection," [Online document], 2002 Nov 8, [cited 2003 Jan 30], Available HTTP: http://home.jwu.edu/jwright/papers/l2-wlan-ids.pdf

[2] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11," Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, 2001 Jul.

[3] S. Fluhrer, I. Mantin, and A. Shamir, "Weakness in the Key Scheduling Algorithm of RC4," 8th Annual Workshop on Selected Areas in Crytography, 2001 Aug.

[4] Snax, "AirSnort Homepage," [Website], 2002 Sep 25, [cited 2003 Jan 30], Available HTTP: http://airsnort.shmoo.com

[5] A. T. Rager, "WEPCrack – An 802.11 key breaker," [Website], [cited 2003 Jan 30], Available HTTP: http://wepcrack.sourceforge.net/

[6] R. Fleck and J. Dimov, "Wireless Access Points and ARP Poisoning," [Online document], 2001 Oct 12, [cited 2003 Jan 30], Available HTTP: http://www.cigitallabs.com/resources/papers/download/arppoison.pdf

[7] 802.11ninja, "802.11ninja.net," [Website], [cited Jan 14], Available HTTP: http://802.11ninja.net/

[8] Air Defense Inc, "Wireless LAN Security for the Enterprise," Air Defense, [Website], [cited 2003 Jan 30], Available HTTP: http://www.airdefense.net/

[9] AirMagnet, "Air Magnet," [Website], [cited 2003 Jan 30], Available HTTP: http://www.airmagnet.com/

[10] Finisar, "Surveyor Wireless," Finisar, [Website], [cited 2003 Jan 30], Available HTTP: http://www.gofinisar.com/index.html

[11] Black Alchemy Enterprises, "Black Alchemy Weapons Lab: Fake AP," Black Alchemy Enterprises, [Online document], 2002 Oct 12, [cited 2003 Jan 30], Available HTTP: http://www.blackalchemy.to/Projects/fakeap/fake-ap.html

[12] J. L. DeBoer, "Digital Matrix – AirSnare," Digital Matrix, [Online document], [cited 2003 Jan 30], Available HTTP: http://home.attbi.com/~digitalmatrix/airsnare/

[13] S. Barber, J. Chung, D. Kimdon, D. Lopes, B. McClintock, and D. Wang, "OpenAP," [Website], [cited 2003 Jan 30], Available HTTP: http://opensource.instant802.com/