A MAC Layer Covert Channel in 802.11 Networks

Ricardo Goncalves
Department of Electrical and Computer Engineering, Naval Postgraduate School
Monterey, California
santana.goncalves@marinha.pt

Murali Tummala
Department of Electrical and Computer Engineering, Naval Postgraduate School
Monterey, California
mtummala@nps.edu

John C. McEachen
Department of Electrical and Computer Engineering, Naval Postgraduate School
Monterey, California
mceachen@nps.edu

***Abstract*—Covert channels in modern communication networks are a source of security concerns. Such channels can be used to conduct hidden communications, facilitate command and control of botnets or inject malicious contents into unsuspected end user devices or network nodes. The vast majority of the documented covert channels make use of the upper layers of the OSI model. In this work, we present a proof of concept on a new covert channel in IEEE 802.11 networks, making use of the Protocol Version field in the MAC header. This is achieved by forging modified CTS and ACK frames. Forward error correction mechanisms and interleaving were implemented to increase the proposed channel's robustness to error. A laboratory implementation of the proposed channel and the results of tests conducted on the proposed channel, including measurements of channel errors and available data rate for transmission, are presented. The results validate the viability of the proposed covert channel and demonstrate that robustness of the channel to frame errors can be improved by using well known forward error correction and interleaving techniques.**

Keywords - IEEE802.11 MAC frame; frame forging; covert channel; protocol version

I. INTRODUCTION

As wireless networks become more ubiquitous, so do our dependencies on them. According to an industry report, in 2012 over a billion devices will be shipped with technology based on this standard onboard and the number is projected to be over two billion in 2014 [1]. Mobility and ease of access of wireless networks are very attractive characteristics to the end users, but along with them come additional security concerns [2].

In order to protect wireless networks from being exploited, we need to constantly evaluate their vulnerabilities and devise techniques to mitigate them. Finding possible covert channels presents an ongoing challenge, and the potential uses for such channels range from well-intentioned authentication mechanisms [3] to malware propagation [4], exfiltration [5] or command and control of botnets [6].

Many covert channels have been documented over the years and reflect the technological stage of the networks at which they were documented. The idea of network covert channels was documented 25 years ago by Girling [7], although the concept of a system-based covert channel was initially presented by Lampson in 1973 [8]. The vast majority of academic research has focused on documenting covert channels in layer 3 (network layer) or above (transport, session, presentation and application layers) of the OSI model [9]. These types of covert channels based on higher layer protocols span a wider variety of networks, since they are not limited by the physical or medium access mechanisms. The two most explored protocols above layer 2 (data link layer) are IP and TCP [10]. Even higher layer protocols, such as ICMP, HTTP or DNS, have several documented covert channels [10].

Recently, researchers began investigating wireless networks, specifically identifying covert channels in the MAC layer [11,12,13]. Frame forging plays a key role in this type of covert channel. Creating fake frames with modified header bits is a recurring theme to implement such channels. MAC header fields such as the sequence number [12], initialization vector [12] or destination address [13], have been used to hide the covert information.

Frikha, et al. [12] proposed two different implementations of a covert channel, both using fields in the 802.11 MAC header. The first one uses the 8 most significant bits of the sequence control field; the second implementation applies to networks that use Wired Equivalent Privacy (WEP) where the initialization vector subfield is used to carry the covert message. Another covert channel, as proposed by Butti [13], uses part of the destination address field of ACK frames to hide the payload. Each of these approaches relies on the forging of frames by manipulating the contents of the MAC header in order to hide the covert information.

In this paper, a covert channel that will use the MAC header of control frames is proposed to hide the covert information. This will be achieved by forging frames that use the protocol version bits in a way that was not intended by the designers of the IEEE 802.11 standard. Specifically, the protocol version field and selected control bits in the MAC header field are used to accomplish this. Our work also addresses the error robustness and throughput of the channel, supported by experimental results.

The rest of the paper is organized as follows. Section II presents an overview of the IEEE802.11 MAC frame fields and an analysis of network frame traffic. The proposed covert channel is described in Section III. Section IV presents the results of experiments.

 ISBN: 978-1-61208-174-8

II. IEEE802.11 NETWORKS AND FRAME TRAFFIC

IEEE 802.11 based wireless nodes share a common medium for communication. The fundamental building block of the 802.11 architecture is called the Basic Service Set (BSS). One BSS may be connected to other BSSs via a Distribution System (DS). Within this framework, stations can connect in ad-hoc mode or infrastructure mode. The simpler case is ad-hoc mode, where two stations can connect directly, point to point, without a DS and an Access Point (AP). If we have the stations connecting via an AP and making use of a DS, then we say they are setup in infrastructure mode.

A. 802.11 MAC frame format

A generic MAC format for an 802.11 MAC frame can be seen in Figure 1. The frame consists of the MAC header, the frame body and the Frame Check Sequence (FCS).

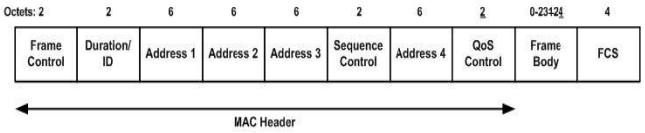

Figure 1. MAC frame format (from [14]).

The first field in the MAC header is the Frame Control (FC), consisting of two octets, and its contents are shown in Figure 2, with the protocol version field highlighted. This field consists of two bits that represent the version number of the 802.11 protocol being used. As of this writing, PV is expected to be set to zero [14]. This value may change in the future if a newer version of the standard is released.

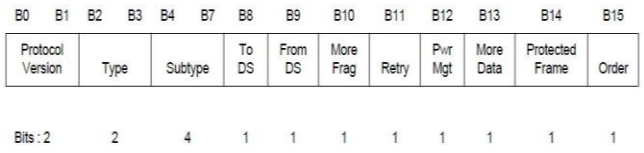

Figure 2. Frame control field (from [14]).

In the proposed covert channel, we utilize the remaining three possible combinations of the PV field to hide the covert information.

B. Frame Types of Interest

Four different types of frames exist in the 802.11 protocol: management, data, reserved and control frames.

Control type frames facilitate the exchange of data frames between stations. Within the existing control subtypes, we are interested in the smaller sized frames, the Acknowledgement (ACK) and the Clear To Send (CTS). These frames also tend to be present in large volume.

The IEEE 802.11 MAC layer makes use of the CSMA/CA scheme, in order to minimize the number of collisions and subsequent frame loss. To address the hidden node problem, a RTS/CTS handshake mechanism is used. The CTS is a 14-byte long frame whereas the RTS is 20 bytes long.

The ACK frame is generated when a station correctly receives a packet, and it is intended to signal the source station that the reception was successful. For this reason, this type of frame also tends to be very common in an operational wireless network. The length of this frame is the same as the CTS, 14 bytes.

Both frames share the same format and they only differ in one bit in the subtype field within the frame control. The ACK frame has the subtype value set to 1101; the CTS sets it to 1100.

C. Network Analysis

A heavily used 802.11 network on campus is monitored to collect frame traffic on multiple channels. From the MAC frame traffic collected, channel 1 is found to be the one with most traffic volume and number of users. We collected over 22 million packets to analyze the following frame basic characteristics.

Ideally, we want a frame that is short in length, common in occurrence, and still valid if some bits are changed. Additionally, its presence in bursts shouldn't be a rare event. These features are desirable for achieving a reasonable throughput while providing covertness.

The results of our analysis are shown in Figure 3 as a pie chart, which represents the frequency of occurrence of different types of frames. The data frames are dominant, followed by CTS, ACK and beacons. The "others" refers to the sum of all other frames that represent less than 1% individually. From this plot we can clearly see that two types of control frames matching our needs stand out, the ACK and the CTS.

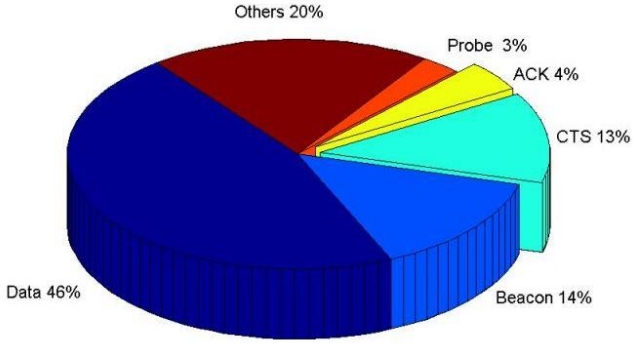

Figure 3. Frequency of occurrence of the monitoried frame types.

D. Choosing the Frame Type

In the process of choosing a frame for the covert channel, several frames were considered, such as RTS and ACK. These frames could serve as well as the CTS, but they were found to be less frequent than CTS. Also, among these three frames, RTS is the longest one with 20 bytes, and the CTS and ACK have only 14 bytes. For this reason we narrowed the options to ACK and CTS.

From monitoring of frame traffic on the campus wireless network and empirical analysis, we found that the CTSs occur with a frequency two times higher than that of the ACKs. The monitoring was conducted in different traffic scenarios, ranging from low traffic periods to high levels of

 ISBN: 978-1-61208-174-8

utilization of the network. We chose to use CTS for building the proposed covert channel as the CTS traffic volume is large and is of same frame size as ACK. By choosing CTS, we can minimize the chance of causing a traffic anomaly based on the type and frequency of packets flowing through the network.

Since CTS and ACK have a similar frame structure, it is easier to switch from one to the other, according to our objectives. The main concept of the proposed covert channel applies equally to both frames. It is even possible to have one end of the channel transmitting ACK frames, and the other transmitting CTS frames, without any loss or degradation of performance. Alternating frame types, such as transmitting a forged ACK followed by a forged CTS is also viable. Many other variations are also feasible.

The fact that both CTS and ACK frames do not contain a source address also contributes to a higher level of stealthiness, since it is not possible to immediately identify the source of the transmission.

III. PROPOSED COVERT CHANNEL

This section describes the proposed covert channel and the use of forward error correction and bit interleaving mechanisms to improve its performance.

A. MAC Header Manipulation

In the proposed covert channel we use two bits in the protocol version field of the MAC header of an 802.11 CTS packet to carry hidden information. The proposed covert channel uses the protocol version bits in a variety of ways to signal the beginning and end of the transmission as well as to carry the information, one bit at a time. A graphical representation of the bits being used is shown in Figure 4.

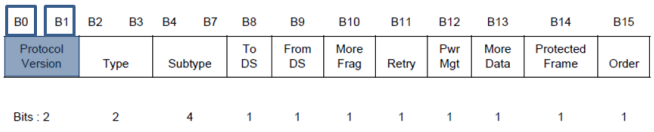

Figure 4. Manipulated bits in Frame control field (blue squares).

In order to facilitate communication in the proposed covert channel, we divided the transmission into three segments: start message delimiter, message, and end message delimiter. The start and end delimiters are realized by transmitting a sequence of five frames with 01 in the protocol version field. The message bits are transmitted using combinations of 10 as binary "0" and 11 as binary "1" in the protocol version field. The message is organized into 8-bit ASCII characters.

B. Forward Error Correction

Since we are operating in a shared media, collisions will eventually occur. This will be interpreted as an error, since a frame carrying covert payload will be lost. To mitigate the effect of frame losses, and thus reduce the number of errors in the covert channel, the use Forward Error Correction (FEC) was considered.

There are several options for implementing FEC: block codes such as Hamming and Reed-Solomon, convolutional codes, turbo codes, or low density parity check codes. In this work, however, a convolutional code was used for error correction.

A convolutional coder takes an m - bit message and encodes it into an n - bit symbol. The ratio m/n is known as the code rate. In our case a code rate of $2/3$ was used, meaning the encoded message will be one and a half times as long as the original message. This will increase the time needed to transmit the same message as before, since a higher number of bits is being sent.

Another important parameter in convolutional coding is the constraint length. This parameter, *k*, represents the number of bits in the encoder memory that affect the generation of the n output bits [15]. A constraint length of 4 is used in our experiments.

Forward error correction is typically applied to a transmission of a stream of bits sent and received sequentially. In our case, however, the bits are embedded into independent frames, which are prone to loss. As a result, when a frame is lost, the receiver has no indication that a bit was missing. Consequently, we now need to know exactly which frames were lost in order to apply the FEC correctly.

One option is to use the eight flag bits in the frame control field of the MAC header to index a longer sequence number, which makes determining the location of lost frames an easier task. These flag bits will not carry any covert information but serve only the error correction function. However, it is important to state that applying this use of the flag bits will increase the probability of detection of the covert channel, since unexpected flag attributions will be present. In this situation, we move from a minimum deviation of two bits (as in Figure 4) to a maximum of 10 bits (as in Figure 5). This presents a tradeoff between detectability and error performance, and the user must exercise the option to choose one over the other as dictated by the application. In order not to use the flag bits one could use the type and subtype fields of the MAC header. The IEEE802.11 standard defines some bit combinations of the subtype field as "Reserved". Exploring these combinations could be an option, although we did not test it.

Figure 5 is a representation of how we accommodated the information and sequence bits within the MAC header.

The blue squares represent our covert channel bits. These bits are used in the same way as before: the first bit (B0) signals the presence of the channel and the second is payload (B1). The red circles refer to the sequence bits, which are placed in the flag bits of the frame control field.

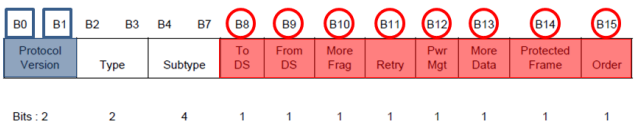

Figure 5. Representation of the frame structure using the flag bits for sequencing (red circles).

 ISBN: 978-1-61208-174-8

Given that we have eight flags, this gives us a total of 256 possible sequence numbers. This alone provides a reasonable amount of protection against a long burst of frame losses, when compared to the previous approach.

C. Forward Error Correction and Interleaving

We now consider sending more than one bit of information per forged frame.

Since each frame now carries more than one information bits, the loss of one or more frames has a bigger impact on the number of errors in the channel. In order to mitigate this effect, we interleave the bit string resulting from the convolutional coder. This consisted of breaking the coded message in blocks of 8 bits, building a matrix with each block in a different row. By reading the matrix out by column, from top to bottom, we generate a new string of bits, effectively interleaving all the 8 bit blocks. The number of rows depends on the length of the message we are transmitting.

Figure 6 is a schematic representation of this idea. At the output of the convolutional coder we interleave the bits in groups of 8 bits. This will result in a new string of zeros and ones, which goes into the covert channel processing block. Here the string is separated in groups of n bits, and each group will become the payload of the forged frames.

Notice that only information bits are encoded and interleaved; in this implementation the convolutional coder is applied after we have the complete message we want to transmit.

Figure 6. FEC and interleaving block diagram.

One possible implementation is to use six bits for payload. The frame is forged as follows: six information bits are placed in the selected flag bits, three other bits are used for sequence numbers, and the first PV bit is set to one, indicating the use of the covert channel. Figure 7 illustrates the proposed structure. The blue squares indicate payload bits, and the red circles are sequence numbers. The green diamond (B0) indicates the presence of the covert channel. Bits B1, B8 and B9 form the sequence number yielding a sequence length of 8. Bits B10-B15 form the payload of six bits to carry the message.

Figure 7. Representation of the frame structure using three bits for sequencing (red squares) and six bits for payload (blue squares).

IV. EXPERIMENTS AND RESULTS

In order to implement the proposed covert channel, we developed the necessary code to forge, transmit, and receive frames. Python was the chosen programming language, due to its simplicity, available libraries and extension modules that facilitated our task. Regarding the OS, a Linux environment was elected, for being more flexible, open source and GNU licensed.

The code is divided into three threads running simultaneously. One thread runs as the receiver, another one as the transmitter, and the third one as a control mechanism in order to handle possible discrepancies in the identification of the beginning and end of the covert communication. Other version 1 frames (with bad checksums) where found circulating in the network, and become noise to our version 1 frames forming the start and end delimiters. Thread3 is responsible for filtering out these unwanted frames.

A. Test bed

Frame traffic was recorded over operational wireless networks, during week days, in order to capture the real-world scenarios.

Three different scenarios were considered and tested. All scenarios consisted of transmitting similar messages during approximately the same time of day. The difference between the scenarios is the way the data was transmitted since we varied the number of payload bits and applied different error mitigation mechanisms.

It is important to notice that stations A and B were operating in the ad-hoc mode, outside the infrastructure wireless network being monitored. The stations transmit without any coordination from the access point. This likely causes collisions, and thus frame losses, which are interpreted as errors for analysis purposes.

A standard sentence was used for all scenarios, with a total of 1408 bytes. In the first scenario, the messages were sent without any error control. The second scenario introduced the use of FEC, and the third used a combination of FEC and interleaving, in order to improve the error robustness of transmitted message. In the following analysis, in order to have a performance benchmark, we used the first scenario as the baseline for comparison with the FEC scenarios,.

B. Results

1) Scenario 1

In Figure 8(a) we can see the profile of the traffic collected for a period of about ten hours on channel 1. Figure 8(b) displays the percentage of errors detected upon reception of the test sentence.

Summarizing this analysis, we observed an average error of approximately 3% for the sentence over a total of 30 sets of transmissions. No error correction or sequencing is at work in this scenario.

 ISBN: 978-1-61208-174-8

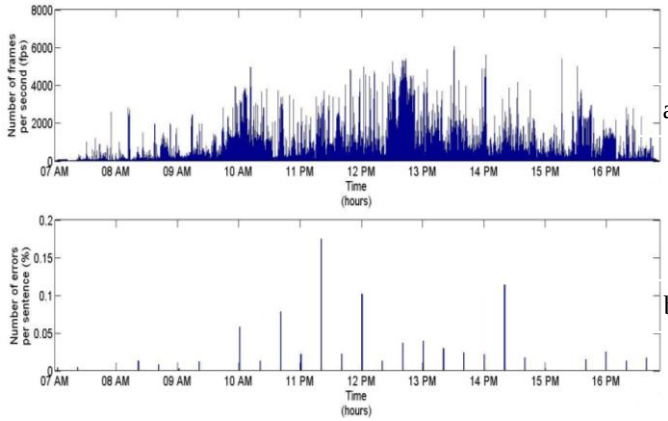

Figure 8. Network traffic profile and percentage of errors for sentence and sequence receptions in channel 1.

2) *Scenario 2*

The percentage of errors as a function of 15 repeated transmissions of the sentence, in channel 1, over a period of 4 hours, is shown in Figure 9. The length of the transmitted sentence is now 2,112 bits long because we applied a $^{2}/_{3}$ rate encoder on a 1,408-bit string. The red stems (x) represent the number of errors detected in the received sentence, and the blue stems (o) the number of errors in the received sentence with FEC. In most cases the number of errors drops to zero or is significantly reduced.

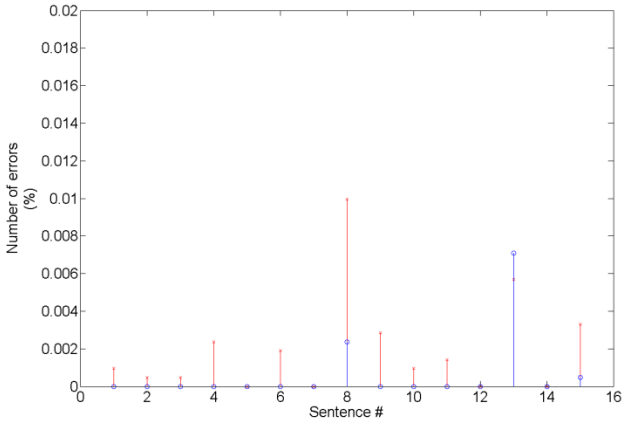

Figure 9. Percentage of erros before (red cross) and after FEC (blue circle) per received sentence, using flag bits for sequencing.

This is consistent with our expectations. We have one outlier in that for the 13th repetition of the sentence we got a higher number of errors with FEC.

We recorded a total of 67 errors in this experiment (without FEC), which translates into an average of 4.5 errors per sentence, or an average error percentage of 0.21%. After the execution of FEC, the total number of errors dropped to 21, resulting in an average of 1.4 errors per sentence or an overall average of 0.09%, relative to the 1,408 bits of the original message. However, this gain was the direct result of having to transmit more bits to send the same message, when compared to the first scenario with no FEC, thus reducing the data rate.

3) *Scenario 3*

The percentage of errors per sentence repetition can be seen in Figure 10. From this figure we can notice an outlier at repetition 12, actually gaining errors after the FEC. This was an isolated event and it was excluded from this analysis. The result is an average number of 1.53 errors per repetition or 0.07% of the total amount of bits sent per sentence. Following the sequence number tracking, de-interleaving and correcting the bit sequence, the total number of errors is reduced to zero. These are significant results; however, the sample space is small, and we cannot conclude that this level of robustness will be achieved in every reception.

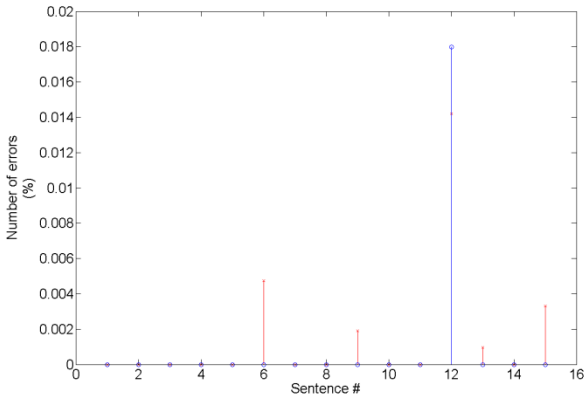

Figure 10. Percentage of erros before (red cross) and after FEC (blue circle) per received sentence with interleaving.

C. *Throughput Analysis*

In order to evaluate the throughput offered in each scenario, the rate at which the frames were transmitted was measured. Being a proof of concept, code efficiency was not a major concern, and the results are presented for analysis purpose only, meaning significant improvements may be easily achieved. This was done using Airopeek and by averaging the rate of the forged frames on a per second (fps) basis. Depending on the network usage at the time, the frame rate varies significantly. Another factor responsible for this variation is the continuous adjustment of the maximum data rate of the network as dictated by the channel conditions. For IEEE 802.11b networks the maximum network data rate possible values are 1, 2, 5.5, and 11 Mbps [14].

To obtain a benchmark for performance comparison, we first determine the maximum data rate possible for the covert channel under optimal conditions. The following conditions are assumed: (i) The channel is ideal with no errors; (ii) there is only one station with frames to transmit; and (iii) we use a data rate of 2 Mbps, the highest possible for 802.11b control frames (basic rate set) [14].

 ISBN: 978-1-61208-174-8

The medium access scheme has to obey some predetermined timing constraints, set by the standard. Figure 11 is a graphical representation of the timing requirements for transmitting a frame.

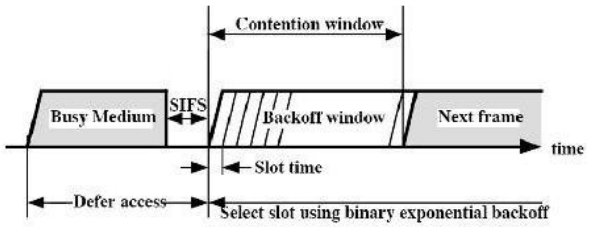

Figure 11. Timing constraints in an 802.11 frame transmission [After 16].

Applying the work of Xiao and Rosdhal [17] to the proposed covert channel, the minimum amount of time necessary to transmit a forged CTS is $t_{min} = 376$ µs, corresponding to a maximum of 2659 forged frames per second. At one bit per frame the maximum bit rate is 2659 bps; at six bits per frame we get 15.954 kbps. The measured throughput values, however, will be significantly smaller.

When we transmit one bit of information in each forged frame, we have an overhead of the start and end delimiters for a total of 10 signaling frames. The measured average frame rate was 61 frames per second. Since each frame represents a bit, and considering our message payload of 1408 bits, we transmit a total of 1418 bits. At 61 fps this corresponds to a total transmission time of 23.25 sec, and a useful bit rate or throughput of 60.5 bits per second (bps).

On the other hand, when we transmitted 6 bits per forged frame and introduced the use of interleaving, the measured average transmission rate was 32 fps. By transmitting a total of 2122 bits, we obtained a total transmission time of 11 seconds. The resulting throughput value is 127.4 bps, considerable improvement over the previous case.

V. CONCLUSIONS

This work presented, implemented and tested a previously undocumented covert channel in an IEEE802.11 network. We used the protocol version field in the MAC header to hide and transfer the covert information. Robustness to errors in the covert channel is improved by the use of forward error correction and bit interleaving. The proposed covert channel was implemented by developing the necessary code in Python. A GUI chat console is used for message transmission. The test bed used for experiments operated in a Linux environment. Preliminary results indicate significant improvement in the error performance of the channel. The achieved throughput of the covert channel is measured and the maximum channel data rate is also determined. The case of 6-bit payload along with convolutional coding and interleaving yielded the highest measured throughput.

REFERENCES

[1] D. McGrath, "WLAN chip set shipments projected to double," in EE Times, 2/17/2011. (accessed March 17, 2011) http://www.eetimes.com/electronics-news/4213260/WLAN-chip-set-shipments-projected-to-double

[2] Y. Xiao, C. Bandela, and Y. Pan, "Vulnerabilities and security enhancements for the IEEE 802.11 WLANs," in Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM) 2005, pp. 1655-1659, 2005.

[3] T.E. Calhoun, R. Newman, and R. Beyah, "Authentication in 802.11 LANs Using a Covert Side Channel," in Communications, 2009. ICC '09., IEEE International Conference, pp. 1-6, 14-18 June 2009.

[4] E. Couture, "Covert Channels," SANS Institute InfoSec Reading Room (accessed January 17, 2011). http://www.sans.org/reading_room/whitepapers/detection/covert-channels_33413

[5] A. Giani, V. H. Berk, and G. V. Cybenko, "Data Exfiltration and Covert Channels," Process Query Systems, Thayer School of Engineering at Dartmouth (accessed February 02, 2011). http://www.pqsnet.net/~vince/papers/SPIE06_exfil.ps.gz

[6] D.T. Ha, G. Yan, S. Eidenbenz, and H.Q. Ngo, "On the effectiveness of structural detection and defense against P2P-based botnets," in Dependable Systems & Networks, 2009. DSN '09. IEEE/IFIP International Conference, pp. 297-306, June 29 2009-July 2 2009.

[7] C.G. Girling, "Covert Channels in LAN's," in Software Engineering, IEEE Transactions, vol. SE-13, no. 2, pp. 292-296, Feb. 1987.

[8] B. Lampson, "A note on the confinement problem," in Communications of the ACM, vol. 16, pp. 613-615, October 1973.

[9] H. Zimmermann, OSI Reference Model, IEEE Transactions on Communications, Vol. COMM-28(4), April 1980.

[10] M. Smeets and M. Koot, "Research report: covert channels," Master's thesis, University of Amsterdam, February 2006.

[11] T. Calhoun, X. Cao, Y. Li, and R. Beyah, "An 802.11 MAC layer covert channel," in Wireless Communications and Mobile Computing, Wiley InterScience (accessed January 2011). http://onlinelibrary.wiley.com/doi/10.1002/wcm.969/pdf

[12] L. Frikha, Z. Trabelsi, and W. El-Hajj, "Implementation of a Covert Channel in the 802.11 Header," in Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08., pp. 594-599, 6-8 August 2008.

[13] L. Butti, Raw Covert (accessed September 2010) http://rfakeap.tuxfamily.org/#Raw_Covert

[14] Institute of Electrical and Electronics Engineers, 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (accessed January 17, 2011). http://ieeexplore.ieee.org

[15] S. Lin and D. J. Costello., Error Control Coding: Fundamentals and Applications, Pearson Prentice Hall, New Jersey, 1983.

[16] W. Stallings, Wireless Communications and Networks, Second edition, Pearson Prentice Hall, New Jersey, 2005.

[17] Y. Xiao and J. Rosdahl, "Throughput and delay limits of IEEE 802.11," in Communications Letters, IEEE, vol.6, no.8, pp. 355- 357, Aug 2002.

 ISBN: 978-1-61208-174-8