A Sliding Window Based Management Traffic Clustering Algorithm for 802.11 WLAN Intrusion Detection

Wenzhe Zhou[1], Alan Marshall[1], ,and Qiang Gu[2]
1 School of Electrical & Electronic Engineering, Queen's University, Belfast, UK
{w.zhou, a.marshall}@ee.qub.ac.uk
2 ECIT, Queen's University, Belfast, UK
qiang.gu@ee.qub.ac.uk

Abstract. This paper introduces a novel Management Traffic Clustering Algorithm (MTCA) based on a sliding window methodology for intrusion detection in 802.11 networks Active attacks and other network events such as scanning, joining and leaving in 802.11 WLANs can be observed by clustering the management frames in the MAC Layer. The new algorithm is based on a sliding window and measures the similarity of management frames within a certain period by calculating their variance. Through filtering out certain management frames, clusters are recognized from the discrete distribution of the variance of the management traffic load. Two parameters determine the accuracy and robustness of the algorithm: the Sample Interval and the Window Size of the sliding window. Extensive tests and comparisons between different sets of Sample Intervals and Window Sizes have been carried out. From analysis of the results, recommendations on what are the most appropriate values for these two parameters in various scenarios are presented.

1 Introduction

Today 802.11 WLANs are widely used in different areas such as home, enterprise, and military due to their convenient nature. However, attacks against 802.11 WLANs are more feasible than against fixed networks as malicious users can easily access the wireless links. Furthermore, research has shown that the security mechanisms such as WEP, WPA and 802.11i provided to protect the WLAN MAC Layer are not completely safe [1] [2] [3]. Even the newly ratified 802.11i standard [4] is vulnerable to certain active attacks such as Denial-of-Service (DoS) and Man-In-The-Middle (MITM) due to the unencrypted management frames and EAPOL (Extensible Authentication Protocol Over Local Area Network) frames [5].

Please use the following format when citing this chapter:

Zhou, W., Marshall, A., Gu, Q., 2006, in IFIP International Federation for Information Processing, Volume 213, Network Control and Engineering for QoS, Security, and Mobility, V, ed. Gaïti, D., (Boston: Springer), pp. 55–64.

Expert security systems are essential in order to address the vulnerabilities of 802.11 WLANs. During the past decade, data mining approaches have been used to improve security because they have the advantage of discovering useful knowledge that describes a user's or program's behaviour from large audit data sets. A good example is intrusion detection, which has been used to protect computer systems as a first line of defense. Next generation Intrusion Detection Expert Systems (NIDES) represents an IDS based on statistics that measures the similarity between a subject's long-term and short-term behaviours [8]. In [9] S.Chebrolu applied Bayesian Networks (BN) and Classification and Regression Trees (CART) approaches in fixed networks to model Intrusion Detection Systems (IDS) with high accuracies of detecting certain intrusions. In [10] C.Kruegel implemented the Bayesian Network approach in IDS to reduce false alarms. To date these approaches have not been applied to implement IDS in the lower protocol layers of WLANs.

Recent research has shown that active attacks exhibit certain fingerprints in terms of the patterns of management frames [6]. By clustering and analyzing certain features of the management traffic (e.g. frame types, source/destination addresses) in the network, an intrusion detection system can recognize and predict the abnormal events. This paper focuses on the clustering of the 802.11 management frames. Our work contributes in three ways: firstly, it specifically addresses the security issues in 802.11 WLANs and uses the characteristic features of management traffic to analyze intrusions; secondly, our algorithm statistically measures the similarity of the management traffic clusters between a long-term and a short-term performance. This meets the requirements of the NIDES systems. Thirdly, we perform a series of test to decide the most appropriate parameters (window size and sample interval) and their optimum values for cluster recognition. This paper is organized as the following: section 2 introduces the background theory and experiment set-up; section 3 introduces the sliding window algorithm for intrusion detection in 802.11 WLAN; section 4 analyzes the choice of parameters for the sliding window algorithm; section 5 gives the conclusion and future work.

2 Background & Challenges

The 802.11 management frames are responsible for the access control, which is expressed by the 802.11 state machine [7]. These management frames are: beacon, probe request, probe response, authentication, association request, association response, reassociation request, reassociation response, disassociation and deauthentication. When stations (STAs) search for connections, join or leave the WLAN, they send out management frames to the access point and negotiate about the access. Research shows that as events (e.g. scanning, joining, leaving, or an active attack) occur in the WLAN, corresponding clusters of management frames are formed [6]. Each event has an individual cluster pattern. By observing the management frame clusters, an IDS can detect the events in a WLAN, and by analysing the pattern of a cluster, the IDS can tell the type of event occurring. In this paper, we focus on how to detect the clusters. The challenges are: how to define the

similarities of the clusters? How to decide the border of a cluster? And how to separate one cluster from another?

We have set up a WLAN with five clients, one attacker and one sniffer to observe the traffic in the WLAN [6]. We injected two types of attacks into the network. Figure 1 shows the complete WLAN traffic trace. The trace file includes the following four events with six clusters: a link problem, a DoS attack, the DoS recovery, and an MITM attack.. From the figure we can see that when events occur, the number of management frames will increase. The "frame density" can be regarded as the number of frames in each time period. From a more abstract point of view, the change of the management frame density shows the occurance of events. Thus we introduce the variance of number of management frames in each sample time point to describe the similarities of the management frame clusters. The average number of frames A can be expressed as:

$$A = \frac{\sum_{i=0}^{n} X_i}{n} \tag{1}$$

And the variance can be expressed as:

$$\delta = \sqrt{\frac{\sum_{i=0}^{n} (X_i - A)^2}{n}} \tag{2}$$

In the above two equations, Xi is the frame number in each sample time point, n is the total frame number during the chosen period of time. The observation of a cluster also includes the contrast between the "past" and the "present" management traffic density in the network. The comparison shows the change in density. The challenges are: (i) how to define the time period of the "past" and the "present"? This can be interpreted as how to decide the value of n in equations (1) and (2) and (ii) how to define the sampling time point. Another challenge is how to describe the comparison with a machine understandable language? In the following section, we introduce our Management Traffic Clustering Algorithm (MTCA), which is based on a sliding window approach.

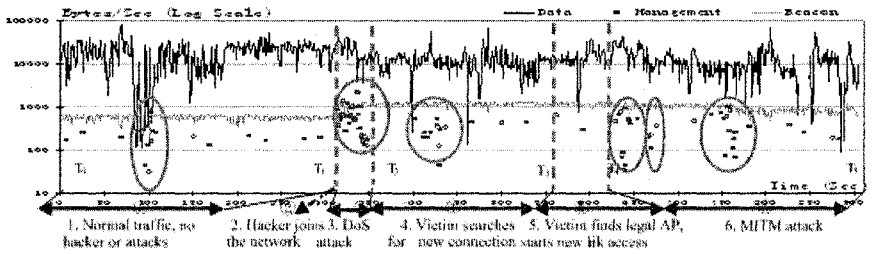

Fig. 1. The complete WLAN traffic trace

3 The Management Traffic Clustering Algorithm (MTCA)

Figure 2 shows the sliding window algorithm. The algorithm contains two parameters: the Window Size (WS) and the Sample Interval (SI). There are two counters calculating the SI and the WS, the SI Counter (SIC) and the WS Counter (WSC). In figure 2, the sliding window body is originally positioned at time t_0. When the SIC counts to the first SI, the sliding window heads forward one SI. At the same time, the original window decreases one SI at the bottom so that the total window size always remains the same length. Thus, the sliding window body moves to a new position at time t_1. As the algorithm keeps running, after n-1 sample intervals, the sliding window will move to new position at time t_{n-1} as shown.

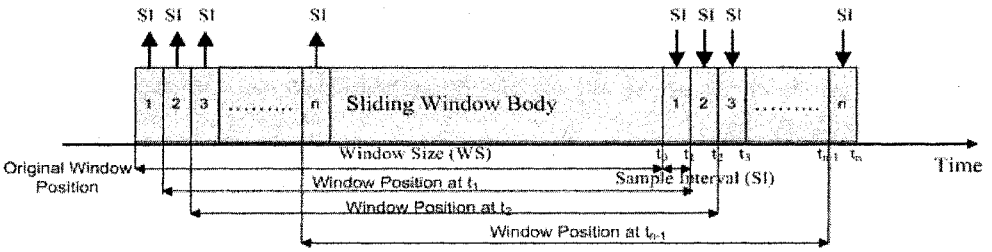

Fig. 2. Sliding Window Algorithm

The sliding window approach meets the requirements discussed in section two for the management frame cluster detection. Firstly, it can easily define the "past" and "present" time periods. We can use the window size WS to represent the "past" time period, its value determines how much importance one wishes to place on past events. The sample interval is used to represent the "present" time period; its value determines how much importance one wishes to place on the present events.

In order to cluster the management traffic in the network, we are concerned with the similarity inside the clusters, that is, the change of the management traffic density.. The change of management traffic density can be represented by comparing

the average density during the time period of the window size and the density during the time period of the sample interval. As described in section two, the average number of frames and the variance of the numbers of frames per time unit can be used to represent the similarity inside of a cluster. We define:

$$WS = K \times SI \tag{3}$$

Where K is the number of sample intervals inside of the sliding window. The value of K decides the how many samples are in the sliding window body. In the following analysis, we use WT_i to represent the total traffic load in the window body at time t_i, ST_i to represent the traffic load in the i^{th} sample interval. For window position at time t_n, we can express the traffic load WT_n as:

$$WT_n = WT_{n-1} + ST_n - ST_{n-1-K} \tag{4}$$

Thus, we can obtain the average number of frames per sample interval AT_n and the variance of the number of frames per sample interval δT_n in the sliding window body at time t_n as:

$$AT_n = WT_n / K = (WT_{n-1} + ST_n - ST_{n-1-K})/K \tag{5}$$

$$\delta T_n = \sqrt{\frac{\sum_{i=n-K}^{n}(ST_i - AT_n)^2}{K}} = \sqrt{\frac{\sum_{i=n-K}^{n}(ST_i - (WT_{n-1} + ST_n - ST_{n-1-K})/K)^2}{K}} \tag{6}$$

We set the sample interval as 0.5s and 20 samples (K=20) in the window body (WS = 10s). From equations (5) and (6), we cluster the management frames using the variance of the number of management frames in each sample interval as shown in figure 3. The management traffic includes: Beacon frame, probe request, probe response, association request, association response, authentication, deauthentication, disassociation.

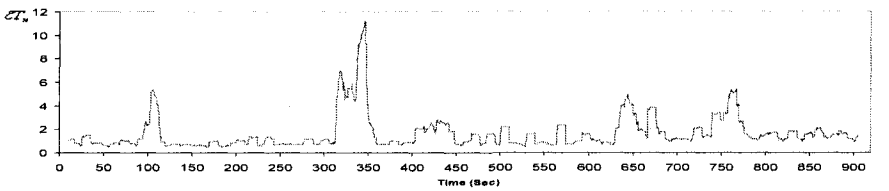

Fig. 3. Management traffic clustering using the sliding window algorithm

The challenge is therefore to detect abnormal events from this trace. Employing a simple threshold scheme can cause erroneous results (e.g. false alarms) because normal events such as the joining of a new client can also cause management traffic clusters, Additionally, some of the abnormal events cause similar clusters to normal events. Also, events can last for unpredictable time periods, for example, the time for a client to joint the network depends on the link quality. Therefore it is also difficult

to base a threshold scheme on the duration of events. An alternative approach is to detect *all* the events occurring in the network and then classify them. In order to do this we need to obtain a discrete distribution of the management frames so that we can regard each separate group as one cluster. We assume that clusters are separated from each other by periods of zero variance.

The beacon frames are continuously sent by the access point. The number of beacon frames and the BSSID field can be used to tell the existence of rogue access points. We can filter out the beacon frames and store the useful information such as the BSSID, the number of beacons, the AP's MAC address in additional space for further cluster recognition purposes. The resultant management traffic can then be clustered as shown in figure 4.

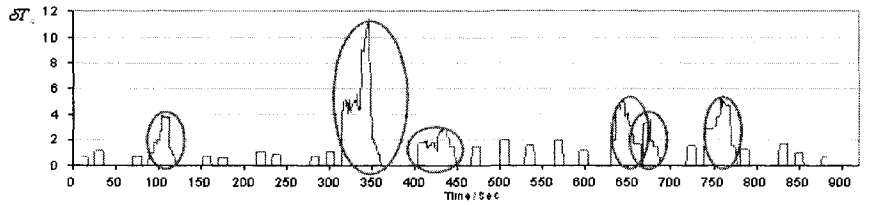

Fig.4. Management traffic clustering without beacon frames

We highlight the six clusters which are supposed to represent the four events. The other groups are normal WLAN events such as clients scanning for new connections. When STAs scan for new connections, they send out probe requests and receive probe responses. The probe frames carry the information about the SSID the STA wish to associate with and the information for synchronization. They don't cause any change to the STA's state machine, but they do provide information to those STAs who wish to search for a new connection. Again, we can filter out the probe frames and also store them as additional information for further cluster recognition purposes, the resultant management traffic clusters are shown in figure 5.

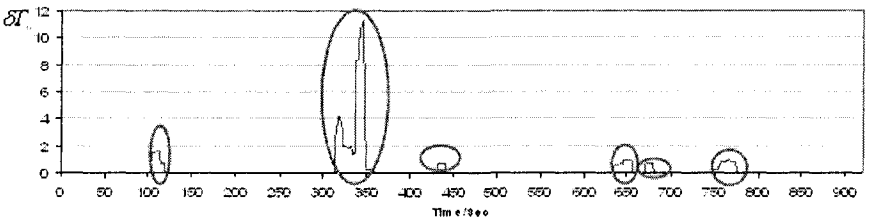

Fig.5. Management traffic clustering without beacon and probe frames

Figure 6 shows the state flow of the MTCA. When the algorithm starts, it transfers to the IDLE state where it remains until triggered by events. When new frames arrive, the algorithm transfers to the FILTER state. If the new frame is a beacon or a probe frame, the FILTER state will transfer to the ADD_storage state

which stores certain information about the frame. Otherwise the FILTER state goes back to the IDLE state. If the time for sample interval expires, the algorithm then goes to the SIFrame Count to accumulate the number of the arriving management frames during the sample interval. If the time for the window size expires, it will transfer to the WS Count state to accumulate the total number of management frames occurring during the window size period. The variance is then calculated and the algorithm returns to the IDLE state.

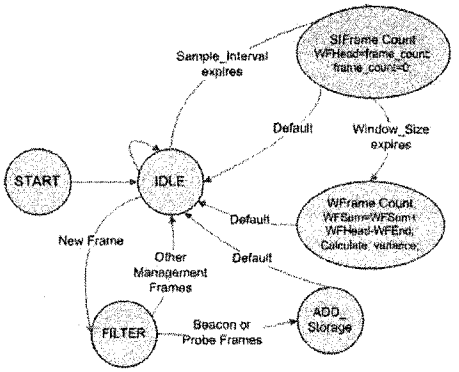

Fig.6. MTCA State Flow

4 Determination of Parameters

There are two parameters in the MTCA: the window size and the sample interval. The value of the WS decides the portion of the "past" traffic the algorithm takes into account, and the value of the SI decides the portion of the "current" traffic. From the point of view of IDS, a small sampling time will introduce many false alarms; a large sampling time will reduce the system's reaction time and thus some events may be missed. The size of the window should satisfy the following constraints:

(i) The window should be long enough so that the algorithm can compare the "past" and the "current" traffic load;

(ii) The window should not be too long so that the "past" part of the traffic will not have an overlarge influence on the "current" traffic.

If we consider the behaviour of abnormal events, the shortest attack can be completed within one second. For example, the deauthentication DoS attack we injected in the experiment sends out on average seven deauthentication frames per second. When the victim received the deauthentication frames, it dropped the connection immediately and started searching for new connections. Yet the attacker continued sending the deauthentication frames so that the victim kept dropping the connection and couldn't get a chance to reconnect with the AP. Thus, we set the sampling time SI less than one second. The choice of window size can be regarded as choosing the value of K, that is, the number of the samples under different sample

intervals. We ran 20 tests for each sample interval with different numbers of samples (K) per interval. Figure 7 shows the number of false alarms generated for each case.

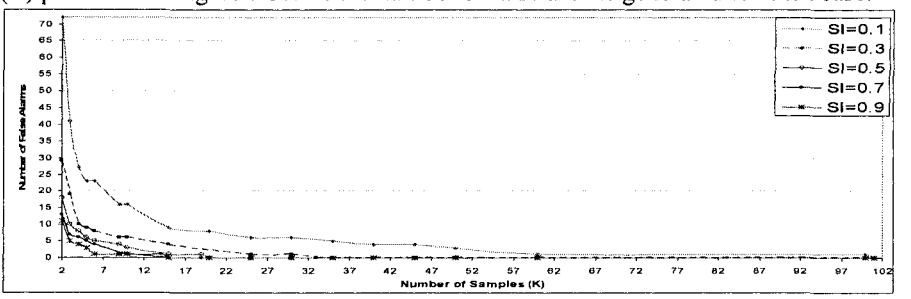

Fig.7. False alarms under different Sample Intervals and number of samples

All the five curves have the same feature: the more samples used, the fewer false alarms are received. This is reasonable since a greater number of samples will generate a longer period, during which more events can be completed. The results also show that larger values of SI reduce the false alarms more rapidly. For SI=0.1s, there are zero false alarms when $K \geq 100$; however, for SI=0.9s, the corresponding zero false alarm point is reached at $K \geq 11$. From the point view of decreasing the false alarms, the larger the sample interval, the fewer false alarms we will get. However a longer sample interval also increases the time to detect events.

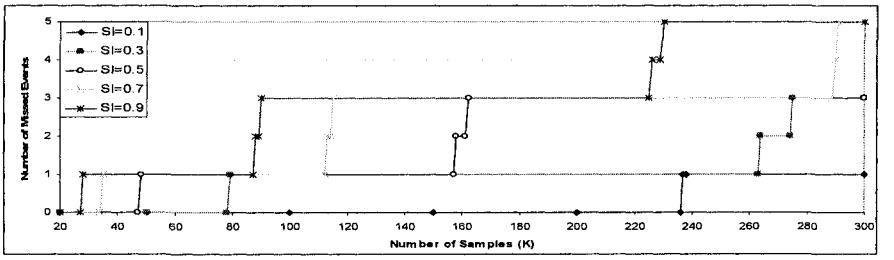

Fig.8. Missed Events under different Sample Intervals and number of samples

The other problem in the choice of parameters is the events which will be missed. Events lasting for a short time period can easily be missed by a large window size and sample interval. We have run more than 100 tests for all five SI values with different numbers of samples. Figure 8 shows the number of missed events under different sample intervals and number of samples. The x-axis starts from 20 which means when the number of samples is less than 20, there are no missed events from any of the sample intervals. When SI=0.1s, there is only one missed event when the number of samples ≥ 237; at the other extreme, when SI=0.9s there is one missed event when the number of samples ≥ 28, and five events are missed when there are 230 samples.

There is therefore a trade-off between the false alarm rate and the probability of missed events. From the point view of security, false alarms are more tolerable than missed events because one missed event could be fatal to the whole system. When choosing the sample interval, we should avoid the 'end' values (i.e. 0.1s and 0.9s); the rest of the choices are SI=0.3s, SI=0.5s and SI=0.7s. For a very busy network which has users frequently joining or leaving it, a smaller sample interval SI=0.3s is preferable since the time between two events is small, a small sampling point would be more accurate for the events; but more false alarms will occur. For a network which is not busy (lightly loaded), we can choose SI=0.7s as we can get less false alarms. The number of samples should be varied depending on the choice of SI. Table 1 shows the range of K for different sample intervals. For SI=0.3s, when K is between 33 and 78, the system will have no false alarms or missed events. For SI=0.5s, when K is between 20 and 47, the system will have no false alarms or missed events; similarly, for SI=0.7s, K should be between 15 and 34. We have also run several sets of tests on other trace files from attacks on both busy and lightly loaded networks. The results showed that the following ranges of K are practical. Generally, when the window size is around 10 seconds, we obtain the best results. From inspection of this we can chose SI=0.5s and the number of samples to be 20, this set of values causes zero false alarms or missed events over a wide range of network loadings.

Table 1. Range of K for different Sample intervals

K	Zero FalseAlarms	Zero MissedEvents
SI=0.3	>=33	<=78
SI=0.5	>=20	<=47

5 Conclusions & Future work

In this paper, we introduce our novel methodology for detecting anomalous events in 802.11 WLANs, the Management Traffic Clustering Algorithm (MTCA). This algorithm is based on a sliding window approach to identify the clusters of management traffic. We use the variance of the number of frames δT_n to represent the similarities inside a cluster. We implemented our theory on a trace file obtained from our experiment described in detail in [6], and obtained a distribution of δT_n. The algorithm filters out beacon frames and probe frames in order to obtain a discrete distribution of δT_n. From the experimental trace six clusters are identified which represent four network events: a link problem, a deauthentication DoS attack, a DoS recovery, and an MITM attack. Analysis of the choice of window size and sample interval for the MTCA is also presented. From consideration of the minimum attacking period, the sample interval should be less than one second. The false alarms and the missed events are two main design criteria. False alarms are more tolerable than the missed events in an intrusion detection system. Nevertheless there is a trade-off between each criterion. From anlaysis of a range of network traces we chose SI=0.3s for a very busy network to avoid missed events and SI=0.7s for a

WLAN which is not busy in order to reduce the of false alarms; pragmatically, we use SI=0.5s as the most reasonable value. For the choice of window size, we obtain the lowest false alarms and missed events when the window size is around 10 seconds.

Future work will focus on the recognition and classification of the complete range of events in 802.11 WLAN. This will involve determining the management frame cluster patterns produced by each type of WLAN attack. The cluster models will then be used to aid real-time analysis and prediction of both normal and anomalous events occurring in WLANS.

References

[1] A.Arbaugh, N.Shankar, J.Wang, and K.Zhang, "Your 802.11 Network has No Clothes", In First IEEE International Conference on Wireless LANs and Home Networks, December, 2001.

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

[3] R.Moskowitz, "Weakness in Passphrase Choice in WPA Interface", http://wifinetnews.com/archives/002452.html, November, 2003

[4] IEEE 802.11i specification. http://standards.ieee.org/getieee802/download/802.11i-2004.pdf

[5] A.Mikhailovsky, K.Gavrilenko and A.Vladimirov, "Wireless Hacking: Breaking Through", December 2004, http://www.awprofessional.com/articles/article.asp?p=353735&seqNum=8&rl=1

[6] W.Zhou, A.Marshall and Q.Gu, "A Novel Classification Scheme 802.11 WLAN Active Attacking Traffic Patterns", Las Vegas, US, WCNC 2006.

[7] IEEE 802.11-1999 specification. http://standards.ieee.org/getieee802/download/802.11-1999.pdf

[8] Debar H, Becker M, Siboni D. "A neural network component for an intrusion detection system." Proceedings of 1992 IEEE computer society symposium on research in security and privacy. Oakland, CA; May 1992.

[9] S.Chebrolu, A.Abraham, and J.P.Thomas, "Feature deduction and ensemble design of intrusion detection systems", Computers & Security, Volume 24, Issue 4, June 2005.

[10] C.Kruegel, D.Mutz, W.Robertson and F.Valeur, "Bayesian Event Classification for Intrusion Detection", in Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC), Las Vegas, NV, December 2003.