Wireless Anomaly Detection Based on IEEE 802.11 Behavior Analysis

Hamid Alipour, Youssif B. Al-Nashif, Pratik Satam, and Salim Hariri, *Member, IEEE*

***Abstract*— Wireless communication networks are pervading every aspect of our lives due to their fast, easy, and inexpensive deployment. They are becoming ubiquitous and have been widely used to transfer critical information, such as banking accounts, credit cards, e-mails, and social network credentials. The more pervasive the wireless technology is going to be, the more important its security issue will be. Whereas the current security protocols for wireless networks have addressed the privacy and confidentiality issues, there are unaddressed vulnerabilities threatening their availability and integrity (e.g., denial of service, session hijacking, and MAC address spoofing attacks). In this paper, we describe an anomaly based intrusion detection system for the IEEE 802.11 wireless networks based on behavioral analysis to detect deviations from normal behaviors that are triggered by wireless network attacks. Our anomaly behavior analysis of the 802.11 protocols is based on monitoring the n-consecutive transitions of the protocol state machine. We apply sequential machine learning techniques to model the n-transition patterns in the protocol and characterize the probabilities of these transitions being normal. We have implemented several experiments to evaluate our system performance. By cross validating the system over two different wireless channels, we have achieved a low false alarm rate (<0.1%). We have also evaluated our approach against an attack library of known wireless attacks and has achieved more than 99% detection rate.**

***Index Terms*— Anomaly detection, IEEE 802.11 security, intrusion detection, wireless network security, protocol analysis, wireless networks.**

I. Introduction

RECENT advances in wireless technology have made it the most widely used communication medium, both in home and enterprise networks. The main advantages of wireless networks versus wired networks are their mobility, flexibility and inexpensive deployment and maintenance cost, especially in places that wiring is difficult. With the exponential growth in the deployment of Wireless Local Area Networks (WLAN), the security issue of these networks has become a major concern for both users and providers.

The first wireless LAN standard, IEEE 802.11, has been ratified in 1997 [1]. Since then, different revisions have been conducted to improve the base standard [2]. Although most of the revisions have focused on the performance, the IEEE 802.11i [3] standard was dedicated to security amendments. Despite IEEE 802.11i has provided good mechanisms to improve privacy and confidentiality, it still does not provide enough protection for availability and integrity (e.g. denial of service, session hijacking and MAC address spoofing attacks) [4].

The failure of current wireless protocols to address these vulnerabilities makes intrusion detection for wireless networks extremely important. The Intrusion Detection Systems (IDSs) can provide more secure networks by monitoring the behavior of the protocol to detect any anomalous events triggered by wireless attacks.

Although different intrusion detection systems are available for wired networks, they cannot be applied directly to wireless networks. While the intrusion detection systems in wired networks are working on different layers of the network, in wireless networks, we cannot easily access the content of the top layers which are often encrypted. Therefore, unlike the wired networks, most of the wireless intrusion detection systems operate at the two lower layers (Physical and Data Link).

We can categorize the available Wireless Intrusion Detection Systems (WIDS) according to the reference data or the analysis techniques. According to the reference data we can group the Wireless IDSs into three groups: those which focus on the physical layer data [5]–[7], those which use the data link layer (MAC layer) data [8]–[10] and the ones which combine the information from both layers [11].

According to the analysis technique, IDSs are classified into two categories: Misuse Detection and Anomaly Detection [12]. In Misuse Detection technique, the system uses a set of rules and signatures to model the malicious activities and attacks. An alarm is generated whenever those rules are triggered or attack signatures are detected. The misuse detection technique is one of the first suggested intrusion detection techniques, and there are many industrial products based on this approach [13]–[15]. The misuse detection systems have low false positive alarms, but on the other hand, they suffer

Manuscript received September 6, 2013; revised April 24, 2014 and December 23, 2014; accepted April 15, 2015. Date of publication May 15, 2015; date of current version August 27, 2015. This work is supported in part by AFOSR DDDAS award number FA95550-12-1-0241, and National Science Foundation research projects NSF IIP-0758579, NCS-0855087 and IIP-1127873. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Vijay Varadharajan.

H. Alipour was with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721 USA. He is now with the Cloud Identity Services and Security Division, Microsoft, Redmond, WA 98052 USA (e-mail: hra@email.arizona.edu).

Y. B. Al-Nashif was with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721 USA. He is now with the Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529 USA (e-mail: yalnashi@odu.edu).

P. Satam and S. Hariri are with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721 USA (e-mail: hariri@ece.arizona.edu; pratiksatam@email.arizona.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2015.2433898

from low detection rate for new attacks especially those of zero day attacks. The other disadvantage of misuse detection systems is that they need to be manually updated with new attack signatures.

In the anomaly detection technique, the system defines a model for the normal behavior of the network and detects any deviation from this normal model as an anomalous behavior. Unlike the Misuse detection, an anomaly detection system with a well-defined normal model can detect new attacks, and there is no need to manually update attack signature library. With better detection performance and no need for manual updates, the anomaly detection is a promising technique, and it is actively pursued by researchers.

In this paper, we present a Wireless LAN (IEEE 802.11) Intrusion Detection System (IDS) based on anomaly behavior analysis of the MAC layer frames with high detection rate and low false alarms. The system builds online models from the state machine transitions of the IEEE 802.11 during its normal operation, and it flags any significant deviation from these state machine transitions as an abnormal activity. To generate the normal transition models, the system extracts any ***n*** consecutive state-machine transitions as an n-gram pattern and stores these extracted patterns as a normal model. At runtime, the IDS system matches the monitored ***n-grams*** of the real traffic sessions with the normal transition model in order to detect any abnormal sessions.

The remaining sections of this paper are organized as follows: In Section II, we discuss the related works. In Section III the general behavior of the IEEE 802.11 protocol is described and rationale behind our approach is discussed. Section IV describes the attacker model and discusses the design of our anomaly detector. The implementation of our wireless intrusion detector is described in Section V, and in Section VI, we present the experimental results and evaluation of our work. The summary of the research presented and planned future work is given in Section VII.

II. Related Works

As it was discussed, wireless Intrusion Detection Systems can be classified based on the reference data and the analysis technique. Our approach is based on anomaly behavior analysis of the MAC layer frames.

Based on reference data, we can categorize the IEEE 802.11 Intrusion Detection Systems (IDS) into three groups: those which focus on the physical layer data [5], [6], [16], [18], [25], those which use the data link layer data [8], [20], [21], [23], and the ones which combine the information from both layers [11]. Most of the physical layer IDS systems check the wireless signal specification in order to detect the physical layer attacks like jamming [5], [18]. The common jamming detection approaches are based on signal strength and location consistency [5], [17]. Furthermore, there are some other works that analyze the physical layer data to detect MAC layer attacks [6], [16], [25]. For example, Sheng et al. used signal strength to detect the MAC address spoofing attack [6].

The MAC layer based systems use the information in the header of the IEEE 802.11 frames for intrusion detection. There is a lot of information in the IEEE 802.11 frames which can be utilized. For example, some approaches use the available sequence number in IEEE 802.11 header to detect the MAC address spoofing [8], [20], [21]. These methods are based on the fact that even if the MAC address can be easily spoofed, the anomalies in the sequence numbers of the spoofed frames compared with the original frames are almost unavoidable by the attackers [22]. They employ data mining and pattern matching techniques to detect these anomalies. Gill et al. proposed a specification based approach to model the state machine of the protocol [23]. They implemented the full state machine of IEEE 802.11 and then monitored the MAC layer frames transferred between each station and access-point in order to generate a state transition model. During testing, any state transition violation is considered as an abnormal activity. Torres et al. [22] combined the sequence number tracking with state machine model to develop a more powerful system. In addition, there are some other approaches which use the combination of physical and MAC layer information to detect the wireless attacks. Fayssal et al. combined the physical and MAC layer information to obtain a richer feature set for their wireless IDS [11].

Although there are different features from different layers that can be used for intrusion detection, adding more features from different layers don't necessarily improve the performance. El-Khatib has applied different feature selection techniques to select the most efficient feature set from IEEE 802.11 header fields [19]. His final feature set includes only 8 features out of 38 possible fields.

According to the analysis type, we can classify wireless intrusion detection systems into two main groups: Misuse Detection and Anomaly Detection. Most of the current commercial wireless intrusion detection systems, e.g. Snort-Wireless [13], AirMagnet [14] and AirDefence [15] are based on the misuse detection approach. There are also some academic works based on misuse detection technique [2]. Since the misuse detection techniques are based on predefined attack signatures, they cannot detect novel attacks, also known as zero day attacks. In addition, the attack signatures should be manually updated whenever a new attack is recognized. The alternative approaches to misuse systems are the ones based on anomaly detection technique. In anomaly based approaches, the normal behavior of the network is accurately modeled, and any significant deviation from the normal model is considered as a suspicious activity. Although the anomaly based systems are able to detect zero day attacks, they often suffer from false positive alarms since any significant changes in the normal traffic might be considered as abnormal operations.

In this paper, we present the design and evaluation of an anomaly based Intrusion Detection System (IDS) for wireless LAN (IEEE 802.11) with low false positive alarms and high detection rates. Our approach is based on anomaly behavior analysis of the MAC layer data. We use the frame type and subtype in IEEE 802.11 header to model the normal IEEE 802.11 behaviors.

The anomaly detection techniques assume that the normal and abnormal events are different and can be identified from

each other, but the challenge is to offer efficient algorithms based on machine learning, pattern recognition and data mining in order to detect the differences between these two types of events. One of the challenges for anomaly based intrusion detection systems is context dependency. By context dependency, we mean that the generated normal models would be biased toward the training context (e.g. traffic volume, training environment, etc.), and might not work well in other traffic contexts (e.g. different network, different wireless channel, etc.). The context dependency of a detection systems increases the false positive error and decreases the accuracy of the system. As we will discuss in detail, our approach mitigates the context dependency issue by splitting the traffic into a group of fine-grained flows, called as ***sessions***, while performing the behavior analysis for each ***session*** separately. The idea is that the normal behavior of the protocol in each session is defined by the protocol state machine and is consistent regardless of the traffic context or intensity.

Another challenge that most of the wireless intrusion detection systems facing is the impact of frame loss on the accuracy of the system. The frame loss is a common issue in wireless networks which can happen due to mobility of the nodes or traffic congestion. As we will discuss in this paper, our technique has a good tolerance against frame loss. The main contributions of our work are the development of an efficient wireless anomaly detection system that overcomes the challenges of anomaly detection algorithms such as high false alarms, context dependency and frame losses.

III. IEEE 802.11 Behavior Analysis

In this section we describe the general behavior of the IEEE 802.11 protocol. Since the IEEE 802.11 is a complicated protocol, here we just go through the related parts which help to clarify our approach.

According to a 2-bit type field in the protocol header, the frames in IEEE 802.11 are grouped into three different types: Management, Control and Data. Each frame group is composed of several different frames which are specified by a 4-bit subtype field.

The data frames encapsulate the data from the top layer protocol in the payload of their frames. The control frames control the access to the wireless medium. They also provide MAC-layer reliability functions. Both data frames and control frames cooperate to deliver the data reliably from one station to another. They work in conjunction to provide area clearing operations, channel acquisition and carrier-sensing maintenance functions, and positive acknowledgment of the received data. Management frames are used to deliver some required primitive supervisory services like *Authentication* or *Association*. They are used to join or leave wireless networks and move associations from one access point to another access point. Unlike the wired networks in which the workstation can be physically authenticated and associated by just plugging it into a switch, in wireless networks these services are provided virtually by using a sequence of management frames. The protocol follows a state machine, shown in Fig. 1, to provide these services.

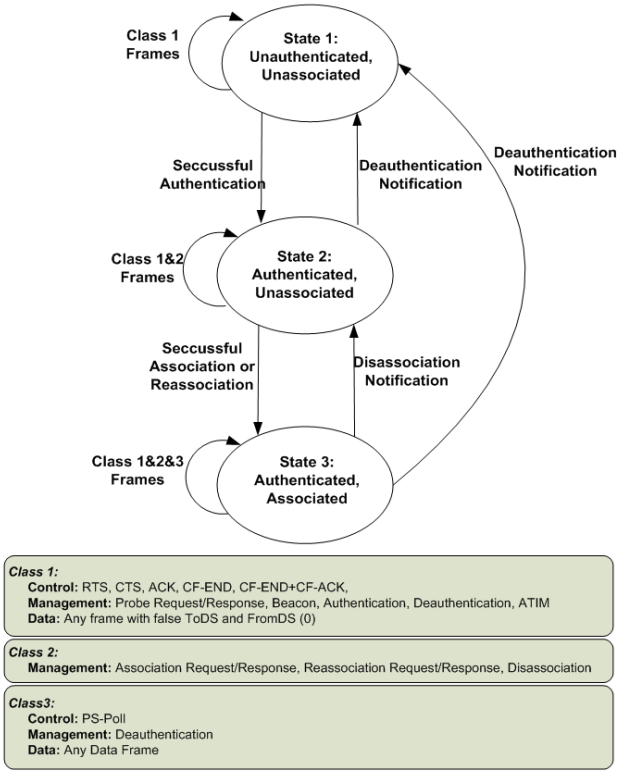

Fig. 1. 802.11 state machine.

Even though the IEEE 802.11i encrypts the payload of data frames to provide more privacy and confidentiality, the management frames are still unprotected, and consequently, they can be easily forged by attackers. There are numerous Denial of Service (DoS) attacks which use this vulnerability to disturb the protocol availability. In most of the attacks with the forged management frames, the attacker violates the normal behavior of the protocol by breaking the aforementioned state machine [17].

Gill et al. have proposed a specification based approach which tracks the protocol state machine in order to detect intrusions [23]. They implemented the full state machine of IEEE 802.11 as an embedded engine. It monitors the MAC layer frames to track the state transition per each *Station* and *AccessPoint* pair. They considered any state transition violation as a potential intrusion. To follow the state transition model, their detection system needs to keep the current state for each connected *Station* to an *AccessPoint* during the connection lifespan. Consequently, the resource consumption of the detection system increases with the number of connected stations. We can consider the resource consumption of their system as O(n) while ***n*** is the number of open connections. The excessive resource consumption can be exploited by attackers who might launch a DoS attacks against the IDS itself. This can be carried out when the attacker keeps opening a lot of new connections (*with or without spoofed addresses using a small network or a large hired bot-net*) and does not terminate them until s/he consumes all the available resources.

Our approaches mitigates this issue by partially modeling the active connections in time interval T instead of keeping the state for all the open connections.

In our approach, instead of defining an explicit state machine using the protocol specification, we partially model the protocol state transitions using ***n-gram*** patterns which are extracted during online protocol behavior analysis. Since we observe the protocol behavior in a specific time window, we just validate the state transitions for that specific time interval and do not keep the state for each connection during its lifetime. So the resource consumption of our system is O(m) while ***m*** is the number of active connections during an observation time window. This significantly reduces the overhead of our approach; we just analyze the active connections during time window T, instead of analyzing all the open connections.

The main advantage of our approach to Gill et al. system [23] is that, unlike their approach which requires some protocol experts manually extract and model the protocol state machine, in our approach the model generation is automatically done by applying machine learning to the protocol normal traffic. This gives more flexibility to our approach in order to be easily extended to other wireless protocols and also let it be easily updated to support new protocol versions. Moreover, as Gill et al. have mentioned, their system is sensitive to frame loss [23]. The main reason of low tolerance against frame loss is that they strictly follow the predefined state machine, while any frame loss can generate an unexpected state transition. In our approach, we generate the normal transition model by observing the normal behavior of the system, which includes the possible normal frame loss scenarios as result of congestion or mobility. In addition, in IEEE 802.11 MAC protocol, a frame will get retransmitted up to a certain number of times if it is lost due to random channel errors. This can generate duplicate copy of the same frames. We use the ***Retry*** flag in the protocol header to filter out and merge this duplicate frames. Therefore, as our evaluation in section VI indicates, our system is tolerant to frame loss scenarios unless there is a dramatic network failure.

IV. IEEE 802.11 Anomaly Behavior Detector Design

The IEEE 801.11 protocol has an explicitly defined state machine that is followed to authenticate and associate a *station* to an Access Point (*AP*). We partially model this state machine transitions in a specific period of time, called *Observation Time Window* ΔT. This method uses a sliding window of fixed size ***n*** to extract ***n-length*** patterns, known as ***n-grams***, from the actual sequence. In this section, we first describe the attacker model, and then will discuss the Anomaly Behavior Analysis methodology and will design a detector for IEEE 802.11 based on our anomaly behavior analysis approach [26], [27].

A. Attacker Model

Our approach is based on partially modeling the IEEE 802.11 state machine to detect the attacks that violate this state machine. The system monitors the protocol state transitions to detect the attack footprints as abnormal transition sequences or excessive repetition of the transition sequences. Referring to the IEEE 802.11 state machine in Fig. 1, it is noticeable that the transitions between different states are triggered by a subset of management frames. Therefore, our anomaly behavior analysis approach aims to detect the attacks against the management frames. The following well-known group of this type of attacks are considered as our attack library in our evaluation in section VI:

1. Injection Test,
2. Deauthentication attack,
3. Disassociation attack,
4. Association Flood,
5. Authentication Flood

These attacks might either generate abnormal transition sequences or excessive repetition of the transitions or sometimes both. For example, disassociation and deauthentication attacks exploit the 802.11 management frames. When a station wants to connect to an AP, it first exchanges authentication frames and then association frames. The connection would be initiated after the station is authenticated and associated. However, the attacker can spoof disassociation or deauthentication frames, pretending to be another station, and keep sending them until s/he receives the acknowledgment from AP that the node has been deauthenticated or disassociated. This anomalous disassociation or deauthentication frames make abnormal protocol transitions during the legitimate station activity.

Although the system is not designed to detect the spoofing attack or attacks against control or data frames, these types of attacks still might be detected as long as they leave one of the aforementioned footprints which means either abnormal transition sequences happen or a sequence of transitions occurs excessively. For example, in the case of spoofing attack three scenarios are possible. In the first scenario, the attacker spoofs the MAC address of a legitimate device that already is active. Since the transmitted frames by the attacker combines with the transmitted frames by the legitimate device as part of the same communication session, they generate abnormal transition sequences which is detected by our system. In the second scenario that usually happens in hijacking attack, the attacker selects an active device and tries some deauthentication or disassociation attacks to disconnect the device from the AccessPoint, and then spoofs the MAC address of the disconnected device to hijack its connection. In this case, though the system does not detect the spoofing itself, it detects the deauthentication or disassociation attack which happens as part of this scenario. The third scenario happens in masquerading attack that the attacker spoofs a different MAC address, which is not active in the network, in order to hide or change its identity. In this scenario, the attacker does not leave any footprints as long as it does not violate the state machine, thus the system cannot detect this case.

B. Anomaly Behavior Analysis (ABA) Methodology

Our anomaly behavior analysis approach is defined over a universe U, which is a finite set of events. U is partitioned into two subsets N and A, which represent the *Normal* and *Abnormal* events respectively, such that $N \cup A = U$ and $N \cap A = \emptyset$. To model the U, we use the representation map R

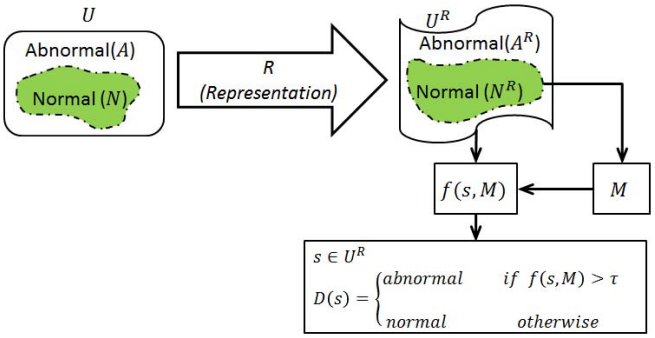

Fig. 2. Anomaly Behavior Analysis Methodology.

which is a function, mapping the events in U to the patterns in U^R as $U \overset{R}{\Rightarrow} U^R$. Likewise, the N^R and A^R respectively represent the *Normal* event set N, and *Abnormal* event set A, such that $N \overset{R}{\Rightarrow} N^R$, $A \overset{R}{\Rightarrow} A^R$ and $N^R \cup A^R = U^R$. A detector is defined as a system $D = (f, M)$ with two components f and M, where f is an anomaly characterization function defined as $f : U^R \times M \rightarrow [0, 1]$ and M is the memory of the system that keeps the extracted normal model from represented normal events N^R. With an output between 0 and 1, function f specifies the degree of abnormality for a sample $s \in U^R$ through comparing it with the stored normal model M. The greater the value of $f(s, M)$, the more abnormality degree for the sample s. The detector D is a binary classifier, which classifies a sample $s \in U^R$ as normal or abnormal by comparing it with M. We can consider D as:

$$D(s) = \begin{cases} abnormal & if\ f(s, M) > \tau \\ normal & otherwise \end{cases}$$

where τ specifies the detection threshold. Detection occurs when the detector classifies a sample as abnormal, regardless of whether it is really an anomaly or a normal sample which is mistakenly classified as anomaly. The detection errors are defined over a test set U_t^R which is a subset of U^R, $U_t^R \subseteq U^R$. Two types of errors are considered for a detector: false positive and false negative. The false positive happens when a normal sample $s \in N^R$ is detected as an abnormal event and is defined as $\varepsilon^+ = \{s \in N^R \mid D(s) = abnormal\}$; the false negative occurs when the detector classifies an abnormal sample $s \in A^R$ as a normal event (undetected anomalies), that is $\varepsilon^- = \{s \in A^R \mid D(s) = normal\}$.

To design a detector D we need to define the following components as it is shown in Fig. 2:

- U: The event set
- R: The representation map
- f: The anomaly characterization function
- M: The Normal model
- τ : The detection threshold

The objective is to design an efficient Anomaly detector for IEEE 802.11 based on protocol behavior analysis which can be achieved by decreasing the detection errors, ϵ^+ and ϵ^-.

C. Detector Design

In our approach, we partially model the protocol behavior by statistically modeling the ***n*** consecutive protocol transitions during a time interval ΔT. By partially modeling the protocol behavior in each time interval ΔT_i, we only analyze the active sessions during that time interval instead of keeping the state of all communication sessions during their lifetime. The intuition behind this technique is that most of the active attacks either show abnormal transition sequences in the protocol or excessively repeat the normal or abnormal protocol transitions. By statistically modeling the sequence of protocol transitions, we can detect the attacks through observing one or both of these footprints.

Based on the formulated problem the following components have to be defined for a detector:

- U: The event set
- R: The representation map
- M: The Normal model
- f: The anomaly characterization function
- τ : The detection threshold

In what follows, we will describe our IEEE 802.11 anomaly behavior detector by defining the aforementioned required components.

The representation map R and anomaly characterization function f are defined based on multiset concept. A multiset is a set of objects with a special property that allows the set to have repeated members. The formal definition of a multiset and its operations is as follows:

Definition 1: A *multiset*, or *bag*, is a set of objects in which the member repetition is allowed [24]. Formally, given a set S, a multiset is defined as $\mathbb{S}(S, c)$, where the function $c : \mathbb{S} \rightarrow Z^+$ counts how often each member $s \in S$ occurs in $\mathbb{S}$. We use $\mathbb{S}(s)$ as shorthand for $c(s)$. We say that s is a *member* of $\mathbb{S}$, denoted as $s \in \mathbb{S}$, if $\mathbb{S}(s) \geq 1$. A multiset is shown by double curly-brackets, for example, $\mathbb{S} = \{\{s_1, s_1\}$ is a multiset over S where $\mathbb{S}(s1) = 2$, and $\mathbb{S}(s) = 0$ for all $s \in S \backslash \{s_1\}$. For a finite multiset $\mathbb{S}$, $\|\mathbb{S}\|$ denotes its *cardinality* and is defined as $\sum_{s \in S} \mathbb{S}(s)$. If $\mathbb{S}_1$ and $\mathbb{S}_2$ are two multisets on S, their intersection is defined as $\mathbb{S} = \mathbb{S}_1 \cap \mathbb{S}_2$ where for all $s \in S$, $\mathbb{S}(s) = \min(\mathbb{S}_1(s)\, \mathbb{S}_2(s))$. Similarly, their *union*, is $\mathbb{S} = \mathbb{S}_1 \cup \mathbb{S}_2$, where for all $s \in S$, $\mathbb{S}(s) = \max(\mathbb{S}_1(s), \mathbb{S}_2(s))$ and their *sum*, is $\mathbb{S} = \mathbb{S}_1 \uplus \mathbb{S}_2$ where for all $s \in S$, $\mathbb{S}(s) = \mathbb{S}_1(s) + \mathbb{S}_2(s)$. The *empty multiset* $\emptyset$ of S is the multiset that $\emptyset(s) = 0$ for all s$\in$ S.

In what follows we will discuss the five steps of how to design the components of an IEEE 802.11 anomaly behavior detector.

*Step 1 (Generating the Event Set **U**):* The event set U for IEEE 802.11 is defined as the set of all possible exchanged frames through the IEEE 802.11 protocol. By investigating the frame format of the protocol [1], we extract two kinds of features: a ***session-key*** and a ***frame-type***. The ***session-key*** feature is used to categorize the frames under different flows of frames called ***sessions*** and denoted as $\boldsymbol{S_1}$. The ***session-key*** for IEEE 802.11 is considered as a pair of addresses which specifies the endpoints of the communication. The ***frame-type*** feature is used to differentiate between different types

of frames. The ***frame-type*** is specified by *type* and *subtype* fields in the IEEE 802.11 frame header.

*Step 2 (Generating the Representation Map **R**):* The representation map R is defined to map the events from U to their representation patterns in U^R as $U \overset{R}{\Rightarrow} U^R$. A communication session S_l can be represented as a sequence of different exchanged frames, $S_l = [\mu_1, \ldots, \mu_k]$, where $\mu_1, \ldots, \mu_k \in$ TYP and TYP is the set of possible values for the frame-type feature. The assumption is that the protocol does not follow a random behavior, and there is an explicit state machine which confirms the normal behavior of the protocol. We consider any n consecutive messages in the session S_l during the time interval $\Delta T_i = [t_i, t_{i+1}]$ as an n-transition pattern called ***n-gram***. To generate the ***n-gram*** patterns, we slide a window of size n over the sequence of the frames in a session exchanged during ΔT_i that is denoted as sub-session $S_{l,\Delta T_i} = [\mu_{j_1}, \ldots, \mu_{j_k}]$. The result is a multiset, or bag, of n-length patterns as:

$$\mathbb{P}_{l,\Delta T_i} = \{\{[\mu_{j_1}, \ldots, \mu_{j_n}], [\mu_{j_2}, \ldots, \mu_{j_{n+1}}], \ldots, [\mu_{j_{k-n+1}}, \ldots, \mu_{j_k}]\}\}.$$

The final representation of U within the time interval ΔT_i, denoted as $U_{\Delta T_i} \overset{R}{\Rightarrow} \mathbb{U}^R_{\Delta T_i}$, is a multiset defined by union of the pattern multisets of all observed sub-sessions in ΔT_i as $\mathbb{U}^R_{\Delta T_i} = \bigcup_{l=1}^{s} \mathbb{P}_{l,\Delta T_i}$ where the observed sub-sessions in ΔT_i are indexed from 1 to s.

*Step 3 (Generating the Normal Model **M**):* To generate the normal behavior model $\mathbb{M}$, we need a normal training set N_T which is a subset of normal events $N_T \subseteq N$ observed during training time frame $T = [t_{start}, t_{end}]$. The normal model $\mathbb{M}$ is considered as the representation of training set: $\mathbb{M} = \mathbb{N}^R_T$ where $N_T \overset{R}{\Rightarrow} \mathbb{N}^R_T$. To apply the representation map R on N_T, first the time frame T is partitioned to k fixed-sized time intervals ΔT_i with size Δt as:

$$\{\Delta T_i\}_{i=1}^{k} = \{\Delta T_i = [t_i, t_{i+1}] \mid t_1 = t_{start}, t_{i+1} = t_i + \Delta t\},$$

where $k = \left\lceil \frac{t_{end} - t_{start}}{\Delta t} \right\rceil$. We apply the described representation map R to the training set during each time partition ΔT_i as $N_{\Delta T_i} \overset{R}{\Rightarrow} \mathbb{N}^R_{\Delta T_i}$ to achieve the normal representation $\mathbb{N}^R_{\Delta T_i}$ within time partition ΔT_i. The final normal model $\mathbb{M}$ is a multiset achieved through union of normal representations of different time partitions as $\mathbb{M} = \mathbb{N}^R_T = \bigcup_{i=1}^{k} \mathbb{N}^R_{\Delta T_i}$. In fact, since this is a multiset union, the final model $\mathbb{M}$ includes the maximum observed count for each pattern in different sessions during different time partitions.

*Step 4 (Generating the Anomaly Characterization Function **f**):* During each time interval ΔT_i, we formulate the anomaly characterization function $f : U^R_{\Delta T_i} \times M \rightarrow [0, 1]$ for each active sub-session $S_{l,\Delta T_i}$ as the complement of the portion of the patterns of the sub-session, $\mathbb{P}_{l,\Delta T_i}$, which have been observed in normal model $\mathbb{M}$. Thus, the f is defined as:

$$f(S_{l,\Delta T_i}, \mathbb{M}) = 1 - \frac{\|\mathbb{P}_{l,\Delta T_i} \cap \mathbb{M}\|}{\|\mathbb{P}_{l,\Delta T_i}\|}$$

where $\mathbb{P}_{l,\Delta T_i}$ is the multiset of the extracted patterns from sub-session $S_{l,\Delta T_i}$. We should mention that the operations $\cap$ (intersection) and $\|\|$ (cardinality) are multiset operations as were defined in ***Defeinition 1***.

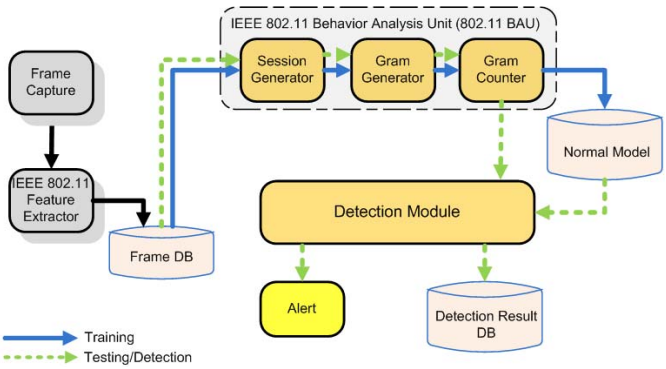

Fig. 3. IEEE 802.11 Behavior Analyzer Architecture.

Step 5 (Tuning the Detection Threshold τ): The detection threshold will be tuned up for the protocol by experimental evaluation of different threshold values to find out which threshold achieves the best detection results with lowest error.

After finding out all the required components, the final detector would be defined over each sub-session $S_{l,\Delta T_i}$ as:

$$D(S_{l,\Delta T_i}) = \begin{cases} abnormal & if\ f(S_{l,\Delta T_i}, \mathbb{M}) > \tau \\ normal & otherwise \end{cases}$$

An alert is issued for each detected abnormal sub-session.

V. IEEE 802.11 Anomaly Behavior Detector Implementation

In our protocol behavior analysis approach, we consider the frequency of a sequence of protocol transitions over a period of time as a measure of whether or not the protocol is behaving normally. During the training phase, state transitions are represented as a multiset of ***n-gram*** patterns ($\mathbb{N}^R_T$), and their statistical properties are captured in the corresponding normal behavior model ($\mathbb{M}$). During the testing phase, the frequency of any N consecutive transitions of the protocol in each session S_l is computed during the observation window ΔT_i as a multiset of ***n-gram*** patterns $\mathbb{P}_{l,\Delta T_i}$ and is compared with the frequency of the normal transitions that are stored in the normal behavior model $\mathbb{M}$. The difference between these two values specifies the anomaly degree for each sub-session $S_{l,\Delta T}$. A General implementation for the designed detector is shown in Fig. 3.

In what follows, we describe how we follow each step of described design to implement the components of an IEEE 802.11 anomaly behavior detector based on the presented architecture in Fig. 3.

*Step 1 (The Implementation of Event Set **U**):* The event set U is generated by the *Frame Capture* and *Feature Extractor* modules, shown in Fig. 3, and the generated events are stored in a database called *Frame DB*.

- *Frame Capture:* This module is composed of a high gain antenna per each monitored channel. Since channel hopping generates frame loss, we use a separate antenna for each channel. The captured frames are delivered to the *Feature Extractor* module for further processing.

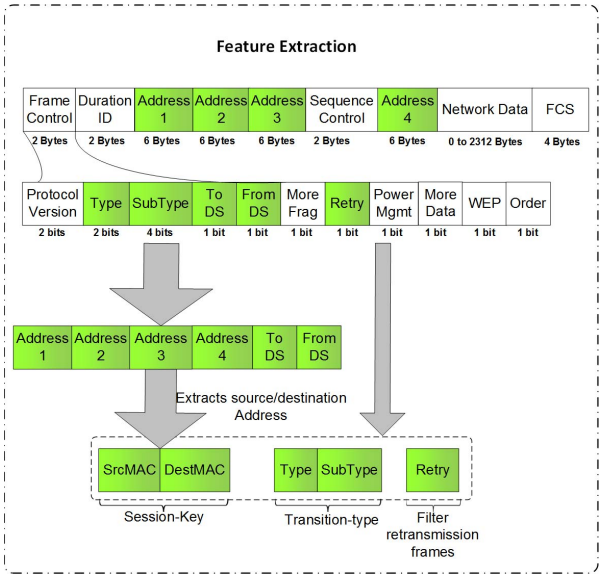

Fig. 4. Feature Extraction Process.

- *IEEE 802.11 Feature Extractor:* This module is used to extract the intended features out of sniffed wireless traffic in order to generate the event set U. As it is described in the design, each event in U will be characterized by a ***session-key*** and a ***frame-type***. Fig. 4 demonstrates how these two features are extracted out of the IEEE 802.11 frame format. The extracted pair of *Source* and *Destination* MAC Addresses is used as ***session-key*** to split the traffic into different sessions, and the *Type* and *Subtype* fields are used as ***frame-type*** for generating the n-gram patterns. Thus, the *Feature Extractor* module extracts the source MAC address (***srcMAC***), the destination MAC address (***dstMAC***), and the type and subtype (***type-subtype***) of the frame. There are four address fields in the IEEE 802.11. The combination of these four address fields with two extra flags (***To DS*** and ***From DS***) specifies the source, destination, receiver and transmitter addresses. Since the approach aims at analyzing the frame flows between endpoint devices, the transmitter address (***Address 2***) and receiver address (***Address 1***) are considered as the ***srcMac*** and ***dstMac*** addresses. In addition, we use the ***Retry*** flag to filter out and merge the duplicate frames which might happen due to retransmission of lost frames. In Fig. 4 the selected fields are colored in green.

*Step 2 (The Implementation of Representation Map **R**):* The representation map R is implemented by the Anomaly Behavior Analysis Unit (BAU). The BAU is responsible to produce the patterns and their statistical information out of the extracted events of event set U. It is composed of the following modules:

- *Session Generator:* The *Session Generator* module classifies the extracted events under different sessions S_l, according to their ***session-key*** (*srcMAC* and *dstMAC* Addresses). All the events with the same ***session-key*** are categorized in the same session denoted as S_l.
- *Gram Generator:* This module slides a window with size n over session S_l during time interval ΔT_i in order to generate the pattern multiset $\mathbb{P}_{l,\Delta T_i}$. It extracts any ***n*** consecutive ***frame-type*** (*type-subtype*) in S_l, as an n-gram pattern.
- *Gram Counter:* The *Gram Counter* module counts the frequency of each ***n-gram*** in time interval ΔT_i per each session S_l, and stores them in normal model.
  There are several DoS attacks which flood the network with malicious frames to interrupt the protocol's operations. This group of attacks indicate an excessive number of normal or abnormal protocol transitions. Thus, by analyzing the statistical properties of the transitions (e.g., frequencies of patterns), the intrusion detection system can detect this type of wireless network attacks. On account of this fact, the ***Gram Counter*** module store the frequency of distinct n-grams that are observed during the observation time interval ΔT_i.

The pseudo codes for different modules of BAU (Behavioral Analysis Unit) are as follows:

```
Session_Generator(frames):
1:   for each Frame f in frames
2:     for each Session s in sessionSet
3:      if (f.sessionKey equal to
                                s.sessionKey)
4:        add f to s;
5:        break;
6:      end if
7:     end for
8:     if (f is not added to any session)
9:       generate a new Session s;
10:      s.sessionKey ← f.sessionKey;
11:      add f to s;
12:      add s to sessionSet;
13:    end if
14:   end for
15:   return sessionSet;
Gram_Generator(sessionSet, gramSize):
1:  for each Session s in sessionSet
2:   win_start ← 0;
3:   if (gramSize greater or equal to
              the number of frames in s)
4:    win_end ← number of frames in s;
5:   else
       win_end ← gramSize;
6:   end if
7:   while (win-end less or equal to
             the number of frames in s)
8:    ngram ← ``'';
9:    for i = win_start to win_end
10:    concat type_subtype of the i^th
         frame in s at the end of ngram;
11:   end for
12:   add ngram to s.ngrams;
13:   increase win_start by one;
14:   increase win_end by one;
15:  end while
16: end for
```

```
Gram_Counter(sessionSet)
1:  for each Session s in sessionSet
2:   for each ngram ng in s.ngrams
3:    if (ng is in s.ngramMultiSet)
4:     increase the count of ng in
                  s.ngramMultiSet by one
6:    else
7:     add ng to s.ngramMultiSet;
8:    end if
9:   end for
10: end for
```

*Step 3 (The Implementation of Normal Model **M**):* The normal behavior model is generated during the training phase indicated by solid blue arrows in Fig. 3. During the training interval T, a subset of normal behavior events known as training data set $N_T \subseteq N$ are fed to the system and the multiset of represented normal ***n-gram*** patterns N_T^R is stored as the protocol normal behavior model $\mathbb{M}$. Thus, the normal behavior model $\mathbb{M}$ contains the observed n-gram patterns during training phase as well as their maximum observation frequencies in a session.

During the training phase, the BAU analyzes the sequence of frames between each *Station* and *AccessPoint* denoted as session S_l to extract the ***n-gram*** patterns per each *Station-AccessPoint* session. Since IEEE 802.11 follows a state machine, it is feasible to assume that after enough training it can extract most of the possible normal patterns in the state machine transitions. To train the system based on supervised learning, some labeled normal wireless traffic should be provided.

The pseudo-code for the normal model generation is as follows:

```
 GenerateNormalModel(frames, gramSize):
1:  define normalModel as a MultiSet;
2:  for each ΔTi partition in T
3:    sessionSet ←
            Session_Generator(frames);
4:    Gram_Generator(sessionSet,
                              gramSize);
5:    Gram_Counter(sessionSet);
6:    for each session s in sessionSet
7:     normalModelDeltaTi ←
        Union_MultiSet(normalModelDeltaTi,
                      s.patternMultiSet);
8:   end for
9:     normalModel ←
             Union_MultiSet(normalModel,
                    normalModelDeltaTi);
10: end for
11: return normalModel;
```

*Step 4 (The Implementation of Anomaly Characterization Function **f**):* The anomaly characterization function is implemented as part of the *detection module*. The *detection module* is used during the testing phase to compare the ***n-gram*** frequencies for each session S_l during time interval ΔT_i with the normal values stored in *Normal Model* $\mathbb{M}$ and computes the anomaly score known as ***a-score***.

We implement the ***a-score*** for each sub-session based on the anomaly characterization function f as:

$$\text{a-score}(S_{l,\Delta T_i}) = f\left(S_{l,\Delta T_i}, \mathbb{M}\right) \times 100$$

$$\text{a-score}(S_{l,\Delta T_i}) = \left(1 - \frac{\left\|\mathbb{P}_{l,\Delta T_i} \cap \mathbb{M}\right\|}{\left\|\mathbb{P}_{l,\Delta T_i}\right\|}\right) \times 100$$

The ***a-score*** is the complement of the portion of the patterns of the sub-session, which have been observed in normal model $\mathbb{M}$. Once the ***a-score*** of a sub-session exceeds the specified threshold, that sub-session is considered abnormal, and the system alerts.

The pseudo codes for the detection module is as follows:

```
Detect(frames, gramSize, normalModel):
1:  sessionSet ←
              Session_Generator(frames);
2:  Gram_Generator(sessionSet, gramSize);
3:  Gram_Counter(sessionSet);
4:  for each session s in sessionSet
5:   normalng = MultiSet_Intersection
          (s.ngramMultiSet,normalModel);
6:   normal_Score ←
     Cardinality(normalng)/Cardinality(s);
7:   a_Score ← 1- normal_Score;
8:   if (a_Score greater than threshold)
9:    issue an alert for s;
10:  end if
11: end for
```

Following are the pseudo-codes for the utilized multiset operations:

```
MultiSet_Union (mS1, mS2):
1: define mUnionResult as a MultiSet;
2: add all elements of mS1 and mS2
                    to the mUnionResult;
3: for each element e in mUnionResult
4:  mUnionResult.count(e) <-
         max(mS1.count(e), mS1.count(e));
5: end for
6: return mUnionResult;
MultiSet_Intersection (mS1, mS2):
1: define mIntersecResult  as a  MultiSet;
2: for each element e in mS1
3:  if (e is in ms2)
4:   add e to mIntersecResult;
5:   mIntersecResult.count(e) ←
       min(mS1.count(e), mS1.count(e));
6:  end if
7: end for
8: return mIntersecResult;
Cardinality (mS1):
1:  set cardinality to 0;
2:  for each element e in mS1
3:   cardinality ← cardinality +
                          mS1.count(e);
4:  end for
5:  return cardinality;
```

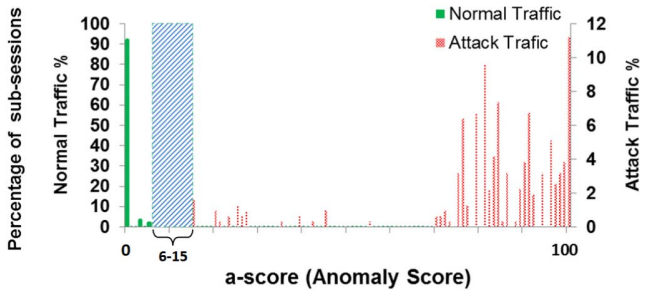

Fig. 5. Comparison of a-score (anomaly score) distribution for normal and attack traffic.

Step 5 (Experimental Tuning of the Detection Threshold τ): To tune up the detection threshold τ for the IEEE 802.11 protocol, we did some experimental evaluation of the ***a-score*** value for normal and attack traffic. Fig. 5 depicts ***the a-score*** (anomaly score) distribution of both attack and normal traffic in the same graph. By comparing the two distributions, we observe that the normal and abnormal traffic can be easily differentiated with a good margin as is specified by blue dashed area. It means the ***a-score*** threshold τ can be set to some value between 6 and 15. A detailed discussion of these evaluation results are presented in section VI.

VI. Experimental Results and Evaluation of Our Approach

To evaluate the performance of our approach, we performed two types of experiments. In the first experiment, we monitored the wireless traffic on two of the high traffic WiFi channels (channel 6 and channel 11) of the ECE department network at the University of Arizona. We used these two traffic sets associated with each channel, CH6 and CH11, to validate the approach. The evaluation is performed by training our intrusion detection system on CH6 (channel 6) and testing it on the other channel, CH11 (channel 11). The validation results have been used to measure the false positive error rate (ϵ^+).

In the second type of experiments, we used a controlled testbed environment to perform a well-known groups of attacks and evaluate the system detection performance under different scenarios. In these experiments, we determined the detection rate of the system which implies the false negative error rate (ϵ^-). The wireless attacks used during the testing phase are discussed later in this section.

A. Training Evaluation

Our training approach is based on supervised learning. It means that we need a comprehensive normal traffic, generated in a supervised environment, in order to create the protocol normal behavior model. Since generating a comprehensive normal traffic is somehow problematic and requires a lot of resources, for the following experiments, training is performed in a semi-supervised environment. To generate a comprehensive normal transitions model of wireless traffic, we have sniffed the wireless traffic on channel 6 (one of the high traffic channels in the ECE department) for one week and have collected around 216 million wireless frames that are stored in the CH6 dataset. Since the training data is not generated in a supervised environment, we have analyzed the collected data to make sure no attack has been included in the training dataset. In order to analyze the dataset, we have generated the n-gram models for the labeled attack traffic and compared the training data with the generated attack models. By analyzing the CH6 dataset, we could not find any connection with more than 5% similarity to the attacks. Thus, it can be assumed that no known attack has happened during the training data collection period.

In addition, since the training data is based on the real traffic, it includes the case of mobile nodes as well. Thus, it is not required to model the mobility scenarios separately. Since the training is performed on a single channel, the mobile nodes are not traced across different channels. This doesn't affect the performance of the system because they are modeled when they are entering the channel (during association) and exiting the channel (during disassociation). In semi-supervised training, though we have assured that the known attacks are not included in the training data, there is no guarantee that no unknown attack (e.g. zero-day attacks) has been included in the training dataset. Since in the following experiments we test the system against a known attack library, using the semi-supervised training does not affect our evaluation. But in a practical deployment it is recommended to train the system using a collected dataset in a supervised environment.

In the following experiments, we have used 4 as the sliding window size and 10 seconds for the observation time window. The intuition behind choosing a window size of 4 is because as our experimental results confirm the 4-gram analysis has provided the best detection and lowest error rates during our evaluation. The other ***n-gram*** sizes are evaluated later in the *Detection Evaluation* subsection. In addition, the 10 second time window seems a good choice because regularly in 10 seconds most of the wireless connection setup process is accomplished, and 10 seconds is an acceptable response time for an IDS (the abnormal behavior can be detected after the 10-second-time-window is processed).

1) Analyzing the Management Frames Case: In this experiment the ***n-gram*** model is only generated out of management frames. If we review the state machine in Fig. 1, we observe that the major transitions between different states are generated only by a subset of the management frames which are defined as set F.

$$F = \{Association\ Request/Response, \\ Reassociation\ Request/Response, \\ Disassociation,\ Authentication, \\ Deauthentication\}$$

Considering this fact, we can significantly reduce the traffic volume to be analyzed. In this experiment, we have selected only the specified management frames from set F to model the state machine.

After testing this approach, we found out that the filtering of the IEEE 802.11 traffic based on management frames doesn't result in much improvement in the detection rate, and in fact it increases the false positive rates ($\epsilon^+ > 7\%$). The reason for this high false positive rate might be the

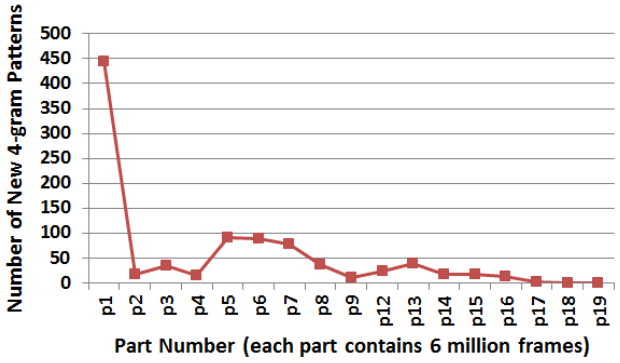

Fig. 6. Number of new observed 4-gram patterns.

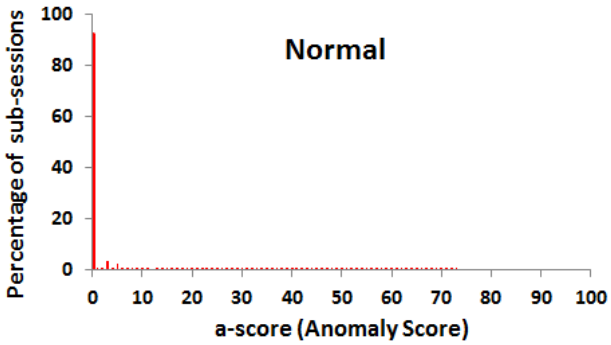

Fig. 7. A-score distribution for normal traffic.

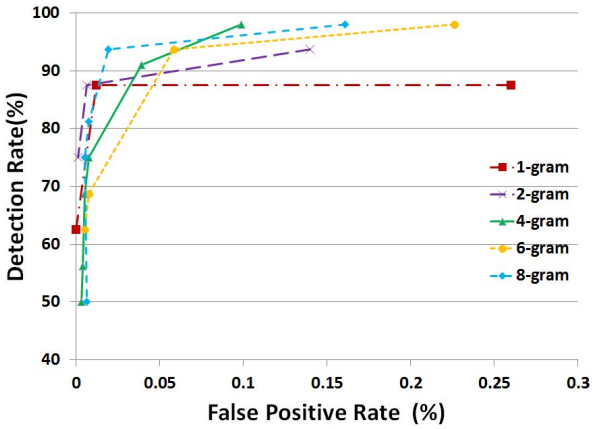

Fig. 8. ROC diagram.

small number of observed management frames. Analyzing the traffic on channel 6, we observed that only 12688 out of these 216 million frames were in the selected set F, which is less than 0.001% of the total traffic. The reason for this small number of these management frames is that these types of management frames are usually exchanged when a new station connects to an AccessPoint or a connected station disconnects or moves from one AccessPoint range to another which happens much less often than the exchanged data or control frames. This small number of frames increases the false positive of the system due to inadequate training data which makes the model biased. In a biased model any minor deviation from training traffic generates a large deviation from the normal model. In addition, it takes a long time to collect enough management frames to achieve an acceptable level of system training. The management frame filtering approach might be still considered in high traffic wireless networks like hotspots, which can result in significant reduction in the number of frames to be analyzed.

2) Analyzing All Frames Case: In this experiment, the normal transition model is generated from all frame types (management, control and data) associated with each session. In this case, the gram generator continues generating ***n-gram*** patterns from the monitored WiFi traffic on channel 6 until we reach the point that the new traffic does not produce any new ***n-gram*** pattern; that means we can expect to have captured almost all possible normal ***n-gram*** transitions. It was noticed that after processing of 102 million frames (p17 in Fig. 6) which were collected after four days, there is no new pattern that can be found in the channel 6 dataset. In this process, we have developed a comprehensive normal transitions model composed of 922 ***4-grams***. Fig. 6 shows the number of observed new ***4-grams*** per each 6 million frames.

To cross validate the approach, we applied the normal transition model that is generated from channel 6 to the observed traffic on the other channel (Channel 11). Fig. 7 shows the ***a-score*** distribution for the normal traffic on channel 11 based on the generated normal transitions model on channel 6. As it indicates, more than 90 percent of the normal traffic on channel 11 has an ***a-score*** close to zero based on the normal transition model generated on channel 6, and almost 100 percent has the ***a-score*** less than 6. This means that if we set the ***a-score*** threshold more than 6, we will have a very small false positive rate on the channel 11. Our test results indicate that we have achieved less than 0.1% false positive on the channel 11. This can be considered as a validation of the context independency of the model which means that by comprehensively analyzing the normal transitions on any channel with enough traffic, that profiling model can model other wireless channels with acceptable accuracy.

B. Detection Evaluation

In this evaluation, we have launched a range of well-known wireless attacks to evaluate the detection capabilities of our approach. In this section, we first define the set of wireless attacks to be used in our evaluation, and then we evaluate the impact of different n-gram sizes on the system performance. We use the ROC (Receiver Operating Characteristics) diagram shown in Fig. 8 to quantify the detection rate corresponding to each false positive.

1) Wireless Attack Library: To evaluate the system against a wide range of wireless attacks we have evaluated our approach to detect a group of known wireless attacks against the IEEE 802.11 management frames. The attacks has been performed in different degrees starting from the minimal effort attacks. By the minimal effort attack, we mean trying the minimum frames that is required to perform a successful attack. For example in the case of Deauthentication attack, according to our analysis, the minimum required Deauthentication frames to successfully deauthenticate a device from the access point was 10, and our system could detect this attack

by assigning the average *a-score* (anomaly score) of 0.75 to it. The description of each performed attack is as follows:

- *Injection Test:* By doing the *Injection Test* the attacker can test if its wireless card can successfully inject frames and also determines the ping response times to the Access Point. The *Injection Test* provides the attacker with valuable information.
- *Deauthentication Attack:* In this attack the attacker sends deauthentication frames to one or more clients who are currently associated with a particular access point to force them to deauthenticate. *Variations: Deauthentication Broadcast, Targeted Deauthentication*.
- *Disassociation Attack:* This is a kind of Denial of Service attack through which the attacker targets the AP with disassociation frames to disassociate one or multiple clients from the network. *Variations: Disassociation Broadcast, Targeted Disassociation*.
- *Association Flood:* This attack is another Denial of Service attack in which the attacker floods the AP with association or reassociation frames.
- *Authentication Flood:* This attack is a type of Denial of Service attack in which the attacker floods the AP with authentication frames.

As we observed in Fig. 5, by comparing both attack and normal traffic in the same graph, we notice that the proposed system can differentiate the normal and abnormal traffic with a good margin specified by blue dashed area. Based on Fig. 5, by setting the threshold τ to some value between 6 and 15 (specified margin by blue dashed area) we are expecting to achieve good detection results.

2) Detection Results: To determine the effectiveness of our approach, we depict the ROC (Receiver Operating Characteristic) curve for different sizes of ***n-grams*** as shown in Fig. 8. In this experiment the system has been trained with the Channel 6 dataset (CH6) and has been tested against the described attack library.

The system is tested with different ***n-gram*** sizes to compare their detection performance. It is predictable that by increasing the size of ***n-gram*** the detection performance of the system is improving, but on the other hand, the false positive rate is increasing at the same time. As it is shown in this experiment, the ***4-gram*** achieves the best tradeoff between the detection rate and false positive rate. The best detection performance of the system was observed with the 4-gram model while the a-score threshold was set to 12. In this case, the system could achieve the detection rate more than 99%, which means the false negative error is less than 1%, ($\varepsilon^- < 0.01$), while its false positive error is less than 0.1%, ($\varepsilon^+ < 0.001$).

In another experiment, we have evaluated the system under different frame drop rates. During this experiment, we have randomly dropped the frames to analyze the system performance under common frame loss situations. As Fig. 9 and Fig. 10 indicate, the system performs stably in situations with less than 70% frame drop rate. After drop rate increases to higher than 70% the detection rate falls and the false positive increases up to 10%.

The reason for the frame loss tolerance is that the system is trained based on normal IEEE 802.11 traffic which includes

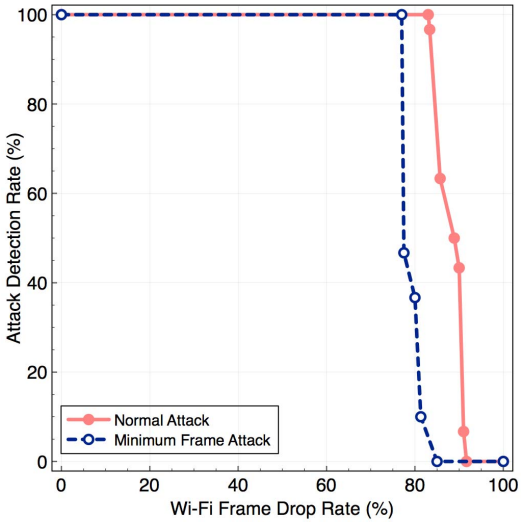

Fig. 9. Detection rate with frame loss.

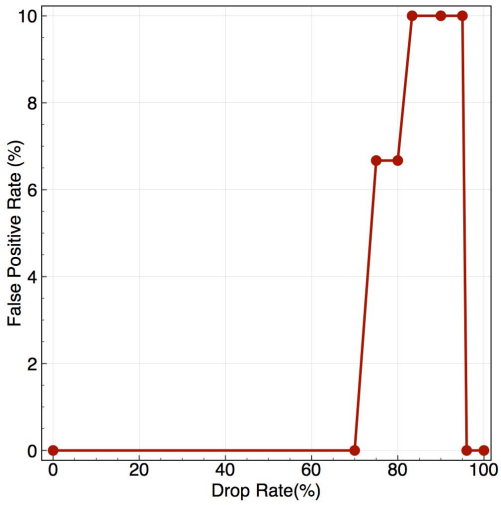

Fig. 10. False positive rate with frame loss.

the normal frame loss scenarios due to congestion or mobility. In addition, in IEEE 802.11 MAC protocol a frame will get retransmitted up to a certain number of times if it is lost due to random channel errors. This can generate duplicate frames. We used the "Retry" flag in the protocol header to filter out and merge the duplicate frames due to frame retransmission. Therefore, a system with adequate training are able to tolerate the normal frame loss scenarios unless a dramatic frame loss happens. In a dramatic frame loss, the frame transmission as well as most of its retransmissions are lost. That is the reason of the noticeable increase in the false positive rate after 70% frame drop. In fact, any dramatic frame loss means the network is broken and should be considered as an anomaly. After 95% both the false positive and detection rate has been dropped close to zero since not much traffic has remained in the network to either be detected or generate false positive.

VII. Conclusion

In this paper, we have reviewed the IEEE 802.11 security issues and briefly reviewed anomaly detection techniques. The main contribution of this paper is that it introduces

an anomaly behavioral analysis methodology and intrusion detection system which can detect different types of IEEE 802.11 attacks with high detection rate (more than 99%) and low false alarms (less than 0.1%). In addition it has been verified that the proposed approach has a good tolerance against frame loss which is a common issue in wireless networks which can happen due to mobility of the nodes or traffic congestion. Our approach is based on supervised learning and anomaly based behavioral analysis technique that builds statistical metrics of fixed size sequential patterns, known as ***n-gram***, to characterize the normal protocol transitions over a period of time during training phase. We have shown that by using ***n-grams*** of size 4 we can accurately detect wireless attacks with less than 0.1% false positive alarms ($\epsilon^+ < 0.1\%$). We have also shown that our approach can accurately detect the known WIFI attacks as: Disassociation Flood, Deauthentication attack, Injection Test, Association Flood, Fake Authentication and Authentication Flood with good false positive rates. We are currently developing other statistical measures that can be used to detect other types of wireless attacks. Since our methodology is based on partially modeling of the protocol state machine, it can be easily applied to other protocols such as Zigbee, Bluetooth, etc.

REFERENCES

[1] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11-1997, 1997. [Online]. Available: http://dx.doi.org/10.1109/IEEESTD.1997.85951

[2] *IEEE 802.11: Wireless LANs*. [Online]. Available: http://standards.ieee.org/about/get/802/802.11.html, accessed Nov. 21, 2011.

[3] *Amendment 6: Medium Access Control (MAC) Security Enhancements*, IEEE Standard 802.11i-2004, Jul. 2004.

[4] C. He and J. C. Mitchell, "Security analysis and improvements for IEEE 802.11i," in *Proc. 12th Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, San Diego, CA, USA, Feb. 2005, pp. 90–110.

[5] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker, "MOJO: A distributed physical layer anomaly detection system for 802.11 WLANs," in *Proc. 4th Int. Conf. Mobile Syst., Appl. Services*, Uppsala, Sweden, Jun. 2006, pp. 191–204.

[6] Y. Sheng, K. Tan, G. Chen, D. Kotz, and A. Campbell, "Detecting 802.11 MAC layer spoofing using received signal strength," in *Proc. IEEE 27th Annu. Conf. Comput. Commun. (INFOCOM)*, Apr. 2008, pp. 13–18.

[7] W. M. Suski, II, M. A. Temple, M. J. Mendenhall, and R. F. Mills, "Using spectral fingerprints to improve wireless network security," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Nov./Dec. 2008, pp. 1–5.

[8] Q. Li and W. Trappe, "Detecting spoofing and anomalous traffic in wireless networks via forge-resistant relationships," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 4, pp. 793–808, Dec. 2007.

[9] M. Raya, J.-P. Hubaux, and I. Aad, "DOMINO: A system to detect greedy behavior in IEEE 802.11 hotspots," in *Proc. 2nd Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2004, pp. 84–97.

[10] S. Radosavac, G. Moustakides, J. S. Baras, and I. Koutsopoulos, "An analytic framework for modeling and detecting access layer misbehavior in wireless networks," *ACM Trans. Inf. Syst. Security*, vol. 11, no. 4, Jul. 2008, Art. ID 19.

[11] S. Fayssal, S. Hariri, and Y. Al-Nashif, "Anomaly-based behavior analysis of wireless network security," in *Proc. 4th Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Services (MobiQuitous)*, Aug. 2007, pp. 1–8.

[12] P. Helman, G. Liepins, and W. Richards, "Foundations of intrusion detection [computer security]," in *Proc. Comput. Security Found. Workshop*, Jun. 1992, pp. 114–120.

[13] *Snort-Wireless*. [Online]. Available: http://snort-wireless.org/, accessed Sep. 5, 2013.

[14] *AirMagnet*. [Online]. Available: http://www.airmagnet.com/, accessed Sep. 5, 2013.

[15] *Airdefence*. [Online]. Available: http://www.airdefense.net, accessed Sep. 5, 2013.

[16] D. Madory, "New methods of spoof detection in 802.11b wireless networking," M.S. thesis, Thayer School Eng., Dartmouth College, Hanover, NH, USA, Jun. 2006.

[17] K. Bicakci and B. Tavli, "Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks," *Comput. Standards Interf.*, vol. 31, no. 5, pp. 931–941, Sep. 2009.

[18] A. G. Fragkiadakis, V. A. Siris, and A. P. Traganitis, "Effective and robust detection of jamming attacks," in *Proc. Future Netw. Mobile Summit*, Jun. 2010, pp. 1–8.

[19] K. El-Khatib, "Impact of feature reduction on the efficiency of wireless intrusion detection systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 8, pp. 1143–1149, Aug. 2010.

[20] D. Dasgupta, J. Gomez, F. Gonzalez, K. Yallapu, R. Yarramsettii, and M. Kaniganti, "MMDS: Multilevel monitoring and detection system," in *Proc. 15th Annu. Comput. Security Incident Handling Conf.*, Ottawa, ON, Canada, 2003, pp. 22–27.

[21] F. Guo and T. Chiueh, "Sequence number-based MAC address spoof detection," in *Proc. 8th Int. Symp. Recent Adv. Intrusion Detection (RAID)*, Seattle, WA, USA, 2005, pp. 309–329.

[22] L. M. Torres, E. Magana, M. Izal, D. Morato, and G. Santafe, "An anomaly-based intrusion detection system for IEEE 802.11 networks," in *Proc. IFIP Wireless Days (WD)*, Oct. 2010, pp. 1–6.

[23] R. Gill, J. Smith, and A. Clark, "Specification-based intrusion detection in WLANs," in *Proc. 22nd Annu. Comput. Security Appl. Conf. (ACSAC)*, Dec. 2006, pp. 141–152.

[24] A. Syropoulos, *Mathematics of Multisets* (Lecture Notes in Computer Science), vol. 2235. Berlin, Germany: Springer-Verlag, 2001, pp. 347–358.

[25] J. Yang, Y. Chen, W. Trappe, and J. Cheng, "Detection and localization of multiple spoofing attackers in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 44–58, Jan. 2013.

[26] H. Alipour, Y. B. Al-Nashif, and S. Hariri, "IEEE 802.11 anomaly-based behavior analysis," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Jan. 2013, pp. 369–373.

[27] H. Alipour, "An anomaly behavior analysis methodology for network-centric systems," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Arizona, Tucson, AZ, USA, 2013.

Hamid Alipour received the B.S. degree in computer engineering from the University of Isfahan, Iran, in 2005, the M.S. degree in computer engineering from Shahid Beheshti University, Tehran, Iran, in 2009, and the Ph.D. degree in computer engineering from the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ, USA, in 2013. He was a Research Assistant with the Autonomic Computing Laboratory and the NSF Center for Cloud and Autonomic Computing, University of Arizona, from 2010 to 2013. He is currently with the Cloud Identity Service and Security Division, Microsoft, Redmond, WA, USA.

His research interests include cybersecurity, cloud computing and cloud security, cyber trust, biometrics and multifactor authentication, behavior analysis, data mining, big data analysis, machine learning, and Internet of Things.

Youssif B. Al-Nashif received the B.Sc. degree in electrical engineering and the M.Sc. degree in communication and electronics engineering from the Jordan University of Science and Technology, in 1999 and 2000, respectively, and the Ph.D. degree in computer engineering from The University of Arizona, in 2008. He was an Assistant Research Professor with The University of Arizona from 2009 to 2014. He has been an Assistant Professor with the Department of Electrical and Computer Engineering, Old Dominion University, since 2014. He is the first Director of the Old Dominion Center for Cybersecurity Education and Research. He is also the Director of the Secure and Trusted Technologies Laboratory. His research interests include cybersecurity, cyber resilience, secure critical infrastructures, cyber trust, cloud security, autonomic computing, data analysis, behavior modeling, and power and performance management.

Pratik Satam received the B.E. degree in electronics and telecommunication engineering from the University of Mumbai, in 2013. He is currently pursuing the M.S. degree with the Department of Electrical and Computer Engineering, The University of Arizona. His research interests include cyber security, computer networks, secure critical infrastructure, autonomic computing, and data analysis.

Salim Hariri received the M.Sc. degree from The Ohio State University, in 1982, and the Ph.D. degree in computer engineering from the University of Southern California, in 1986. He is currently a Professor with the Department of Electrical and Computer Engineering, The University of Arizona, and the Director of the NSF Center for Cloud and Autonomic Computing. His current research focuses on autonomic computing, cybersecurity, cyber resilience, secure critical infrastructures, and cloud security.