Securing the 802.11 MAC in MANETs: A Specification-based Intrusion Detection Engine

Christoforos Panos, Ioannis Stavrakakis
Department of Informatics & Telecommunications
University of Athens
Athens, Greece
cpanos@di.uoa.gr, ioannis@di.uoa.gr

Platon Kotzias, Christos Xenakis
Department of Digital Systems
University of Piraeus
Piraeus, Greece
mte0912@webmail.unipi.gr, xenakis@unipi.gr

Abstract—**Specification-based detection engines share the advantages of signature-based and anomaly-based detection, since they can detect unknown attacks, without the side effects of high rates of false positives. However, such solutions for MANETs have seen limited use. This paper introduces a specification-based detection engine that is built upon the functionality and limitations of the 802.11 MAC protocol, expanding the detection range of such engines in MANETs. The proposed detection engine is deployed at each node and performs detection using a set of specifications, which describe the correct operation of the MAC protocol operating at the host node. The proposed engine introduces a number of significant advantages since it can effectively detect both known and unknown attacks in real time and with minimum overhead. Moreover, it is resilient to the dynamic topologies that are common in MANETs and its deployment requires no protocol modifications.**

Keywords-intrusion detection system; IDS; specification-based intrusion detection engine; mobile ad hoc networks; MANET security; data-link layer attacks; 802.11 vulnerabilities; security vulnerabilities

I. Introduction

The wireless - mobile nature of MANETs in conjunction with the absence of access points, providing access to a centralized authority, make them susceptible to a variety of attacks. An effective way to identify the occurrence of such attacks is through the deployment of Intrusion Detection Systems (IDSs). An IDS utilizes one or more intrusion detection engines, which can be classified into three main categories [1]: (i) signature-based engines, which rely on a predefined set of patterns to identify attacks; (ii) anomaly-based engines, which rely on particular models of nodes' behavior (i.e., normal profiles) and mark nodes that deviate from these models as malicious; and (iii) specification-based engines, which rely on a set of constrains (i.e., description of the correct operation of programs and protocols) and monitor the execution of programs and protocols with respect to these constraints.

Signature-based engines offer low rates of false positives, but they are not effective against new types of attacks, which are not included in the signatures' databases. Furthermore, maintaining and updating a signature database in a MANET environment is difficult to achieve. Anomaly-based detection engines are capable of detecting unknown attacks and alleviate the need for a signature database. However, they typically require the use of complex detection algorithms, generate high percentages of false alarms, and their performance is further reduced when dynamic network conditions (such as high node's mobility) occur. Specification-based engines can detect both known and unknown attacks. Moreover, they avoid high rates of false alarms, since they do not rely on normal profiles, as happens in anomaly detection. In general, the development of specifications for a specification-based engine might be a lengthy and convoluted process, since the developer has to determine what is the expected behavior of each individual application or protocol, and then establish constrains that characterize this behavior. However, this is not the case of the 802.11 MAC protocol, which incorporates a well-defined functionality and parameters that are involved in its operation.

Currently, specification-based engines for MANETs have seen limited use, focusing mainly on detecting attacks that target the network layer. Tseng et al. [2], Hassan et al. [7], and Huang et al. [5] have proposed specification-based engines that monitor the Ad-Hoc On Demand Routing protocol (AODV) [8] for attacks, while Orset et al. [6] and Cheng et al. [3] have focused on developing specifications for the Optimized Link State Routing protocol (OLSR) [9]. These engines monitor only the particular routing protocols, and thus, they are only capable of detecting attacks that target the routing mechanisms of MANETs. Song et al. [4] have proposed a specification-based engine that monitors the Dynamic Registration and Configuration Protocol (DRCP) [10] for attacks that alter configuration information in MANETs. Finally, Raya et al. [12] have proposed a mechanism for detecting misbehaviors in wireless networks relying on the 802.11 MAC [11] protocol. However, this mechanism requires the existence of trusted access points that execute it and monitor connected nodes for malicious behaviors. Furthermore, this mechanism relies on thresholds to detect several attacks, where malicious nodes may attempt to exploit them by performing sporadic attacks considering not exceeding the threshold values and raising alarms.

In this paper, we propose a specification-based detection engine that is built upon the functionality and vulnerable points of the 802.11 MAC protocol. The engine is deployed using a generic, host-based IDS architecture, where each node

This work has been supported in part by the European Commission under the Network of Excellence in Internet Science project EINS (FP7-IST- 288021)

implements an instance of the engine. The engine performs detections using a set of specifications, which describe the correct operation of the MAC protocol operating at the host node. These specifications are expressed through the use of a Finite State Machine (FSM). Each state of FSM corresponds to either a legitimate or malicious behavior of the monitored node and a transition from one state to another is triggered by the node's operations and actions. The proposed engine introduces a number of significant advantages (elaborated in detail in section 4), since it can effectively detect all the types of attacks (both known and unknown) that occur at the 802.11 MAC layer in real time and with minimum overhead. Moreover, it is resilient to the dynamic network topologies that are common in MANETs and its deployment requires no protocol modifications.

The rest of this article is organized as follows. Section 2 analyzes the functionality of the 802.11 MAC protocol, focusing on its vulnerable points. Section 3 presents and analyses the proposed detection engine. Section 4 evaluates the engine elaborating on its advantages and limitations. Finally, section 5 contains the conclusions.

II. The 802.11 MAC Protocol

A part of the data-link layer, the 802.11 Medium Access Control (MAC) protocol is responsible for the coordination of transmissions on a common communication medium and thus plays an integral role in one-hop connectivity between neighboring nodes. A malicious node(s) could modify or circumvent standard protocol operations in order to gain an advantage over network resources or hinder the operation of a MANET, since the affected set of nodes may become unable to communicate with neighboring nodes [16]. In order to safeguard the 802.11 MAC protocol, we first analyze its functionality, security requirements and possible attack methods. The aim of such analysis is to develop a set of specifications that capture the correct operation of a node carrying out the aforementioned protocol. These specifications will be utilized by the proposed detection engine in order to distinguish legitimate and malicious behavior.

The 802.11 MAC protocol supports two different MAC schemes, the Distributed Coordination Function (DCF) and the Point Coordination Function (PCF). In DCF, nodes have to check if the channel is clear before transmitting any data, while in PCF, a point coordinator (i.e., access point) is responsible for coordinating which node to transmit next. We will be focusing on the DCF scheme since in MANETs; there are no centralized access points to act as point coordinators.

In DCF all nodes with data to transmit have an equally fair chance of accessing the network. DCF employs a Carrier Sense Multiple Access with Collision Avoidance access method (CSMA/CA) used to minimize collisions (supplemented by the 802.11 RTS/CTS mechanism). In this method, when a node wants to transmit data, it initiates the process by sending a request to send (RTS) frame, and the destination node replies with a clear to send (CTS) frame. Any other node receiving the RTS or CTS frames retreats from transmitting any data for some random backoff time. In order to prioritize access to the wireless medium, DCF specifies three time windows, the Short InterFrame Space (SIFS), the DCF InterFrame Space (DIFS) and the Extended InterFrame Space (EIFS). The DCF scheme is designed based on the assumption that all of the participating nodes are well behaved and thus, it does not incorporate any security precautions. Furthermore, the implementation of the MAC protocol is done in software rather than hardware and thus, modification of the network parameters is possible [11]. When a node wants to transmit data, one of the following three scenarios takes place.

A. First scenario: the wireless medium is idle

The node will first sense the network for a DIFS period of time in order to resolve if the medium is idle. If it stays idle for such a period, the node can send an RTS message to the receiver. The receiver then has to reply with a CTS message, after sensing the network for a SIFS period of time. RTS/CTS messages inform any node that hears the exchange, the duration of the entire data frame transmission. A node that overhears the exchange of RTS/CTS messages has to adjust its network allocation vector (NAV) field, which indicates the amount of time that the node should wait before sensing the medium again. Finally, a receiver that successfully receives a DATA frame has to reply with an Acknowledgement (ACK) message.

A malicious node may take advantage of RTS/CTS messages in order to misbehave. Firstly, it can assert larger transmission duration while advertising its RTS packet, gaining an advantage in recapturing the channel over other well-behaved nodes (i.e., NAV attack). Another way to gain advantage in recapturing the channel is by decreasing the DIFS parameter, thus waiting less than other nodes before sensing the channel. Finally, a malicious node might frequently transmit RTS/CTS packets even without the need to send any data, resulting in a denial-of-service attack.

B. Second scenario: the wireless medium is busy

The CSMA/CA also adopts a backoff mechanism to resolve channel contention. A node that wishes to transmit and senses that the channel is busy during the DIFS period will use its backoff mechanism. After the expiration of the NAV period of time, the transmitter has to sense the medium for DIFS time plus a random backoff time from the range of [0, CW], where CW is the contention window size, maintained by each node. The backoff timer decreases as long as the medium remains idle. If the channel becomes busy during the backoff time, the node must sense the channel for an additional DIFS/EIFS time, before it can decrease the backoff timer again. Once the backoff timer reaches zero, the node can transmit RTS to the receiver and follow the standard procedure. To guarantee fair access to the shared medium, a node that has just transmitted a packet and has another packet ready for transmission must perform the backoff mechanism before initiating the second transmission.

The backoff mechanism can be manipulated by a malicious node, since the later may choose not to comply with protocol rules and select a small backoff interval in order to gain a significant advantage in capturing the channel over well-behaved nodes. The manipulation of the backoff mechanism can be achieved in the following ways:

- Alter the CWmin value in order to select a CW value from a smaller span of values.
- Avoid doubling the CW value after every collision (see sect. 2.3).
- Bypass the random selection of the CW value and force a smaller CW value.

C. Third scenario: No CTS from receiving node

If the transmitter sends an RTS message but does not receive the corresponding CTS, then the protocol assumes that a collision has occurred at the receiving end. In this scenario the transmitter will perform the backoff mechanism after doubling its CW, thus choosing a value between [0, 2CW]. CW duplication occurs until CW reaches a maximum value CWmax and remains until the packet is finally transmitted or discarded. In both cases CW is reset. Finally, the transmitter follows the same procedure if an ACK message is not received.

A malicious node at the receiving end can disrupt the normal communication by forcing a victim node to retransmit its data resulting in delays, communication overhead, and resource consumption. The malicious can achieve such an attack by avoiding the transmission of CTS or ACK messages. In this case, the sender node (i.e., victim) assumes that collisions have occurred and thus, double its CW, executes the backoff mechanism and then attempt to retransmit its DATA packet. Apart from imposing delays in the victim node, the malicious node (i.e., by dropping a percentage of CTS/ACK messages) may gain a significant advantage in capturing the channel, because adjoining nodes are forced into the backoff stage. This behavior can also disrupt the route discovery process; forcing packets through non-optimal routes. If the malicious node continually forces a transmitting node to timeout its transmission, the later will eventually drop the data packet and report a link breakage to the network layer.

III. The Proposed Specification-based Detection Engine

The proposed specification-based detection engine is deployed using a generic, host-based IDS architecture, where each node implements an instance of the engine. The engine performs detection by monitoring the activities that take place locally at the 802.11 MAC protocol of the hosting node and comparing them with the pre-defined set of specifications. This can be achieved by implementing the detection engine at the network interface card driver level (see fig. 1). The implementation can be performed by utilizing a number of APIs such as the Network Driver Interface Specification (NDIS) [13] interface or the Open Data-Link Interface (ODI) [14], depending on the platform in which the detection engine will be deployed.

The pre-defined set of specifications is built upon the functionality and vulnerable points of the 802.11 MAC protocol, which are analyzed in section 2. In order to present them, we use an FSM. Formally, specifications are defined as a *tuple* (S, NO, S_0, δ, F), where S is the set of all possible states; NO is the set of node operations; S_0 is the initial state; δ is a function that maps node operations from a previous state to the current state; and F is the set of final states that correspond to malicious behaviors. In the remainder of this section we exemplify the specifications that describe the correct operation of the 802.11 MAC protocol. In order to simplify the presentation of the proposed engine, we divided the specifications into three set according to the node's communication condition: (a) idle, (b) transmitting, or (c) receiving data. Nevertheless, the detection engine can be expressed inclusively as one FSM.

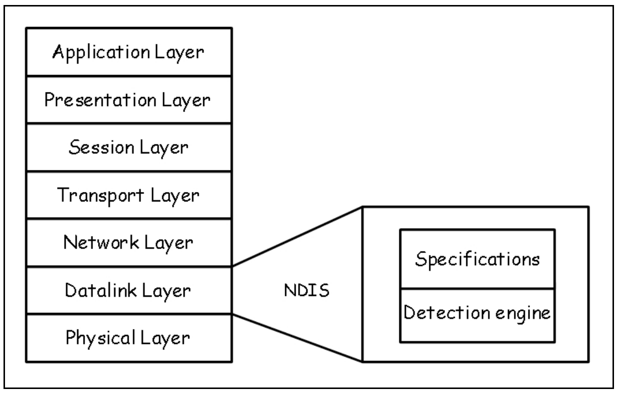

Figure 1. Architecture of the proposed detection engine.

A. Idle node specifications

These specifications describe the operation of the 802.11 MAC protocol when the monitored node is in an idle condition (i.e., the node is not receiving or transmitting any packet). It is presented for the sake of completeness since at this condition, there are no final states designating malicious behaviors. The engine initializes at state S_0 and begins monitoring the host node for any new packets that are ready for transmission or for incoming RTS packets. If the monitored node assembles a packet for transmission, then the engine moves to S_1 (see sect. 3.2). On the other hand, if an RTS packet is received by the node then, the engine moves to S_2 (see sect. 3.3).

B. Transmitter specifications

These specifications exemplify the operation of the 802.11 MAC protocol when the monitored node is attempting to transmit data to another node. The engine starts at state S_1, while the host node has assembled a new packet, which is ready for transmission. At this state, the engine checks if the communication channel is idle or busy. If the channel is idle, then the engine moves to S_3; otherwise, if the channel is busy, the engine moves to S_{12}. In S_3, the expected behavior of the protocol is to transmit an RTS packet (see sect. 3.2.1), while in S_{12} the protocol must call the backoff mechanism (see sect. 3.2.3). An attempt to transmit any data when the channel is busy leads to the final state S_4, which designates a malicious behavior.

1) RTS specifications

The RTS specifications exemplify the correct operation of the protocol during the transmission of an RTS packet by the monitored node. As illustrated in figure 2, the engine first retrieves the DIFS parameter from the physical layer and remains at S_3 until the DIFS timer, feed by the DIFS parameter, expires. If the node attempts to transmit an RTS before the expiration of DIFS, the engine reaches the final state S_5, which designates a malicious behavior. Otherwise, it moves to S_6. In this state, the engine checks if the frame duration field advertized by the RTS packet corresponds to the actual size of the data to be transmitted. If not, the engine moves to the final state S_7, which designates a malicious behavior. Otherwise, the monitored node transmits the RTS packet and the engine moves to S_8. At this state, the monitored node has successfully transmitted an RTS and awaits for a CTS (see sect. 3.2.2).

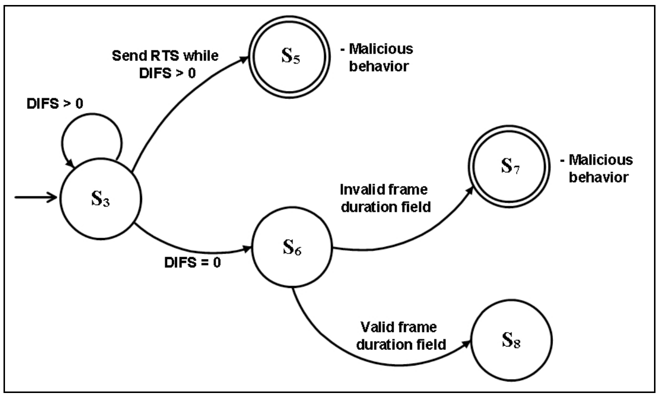

Figure 2. RTS specification diagram.

2) *CTS specifications*

The CTS specifications exemplify the correct operation of the protocol during the reception of a CTS packet by the monitored node. As we can see in figure 3, while in S_8, the engine monitors for incoming CTS packets. If a CTS packet is received before the CTS timer expires, the engine moves to S_9. In this state, the engine retrieves the SIFS parameter from the physical layer and remains at this state until the SIFS timer, feed by SIFS parameter, expires. Subsequently, the engine checks for the transmission of the actual data by the monitored node. If the node does not transmit any data or attempts to transmit them before the SIFS timer expires, then the engine moves to the final state S_{10}, which designates a malicious behavior. Otherwise, it moves to S_{11}, which is the last state of the CTS specifications. The engine remains at this state and awaits for an ACK packet, until the ACK timer expires or the ACK packet is received. Then, it moves to S_{12} (see sect. 3.2.3).

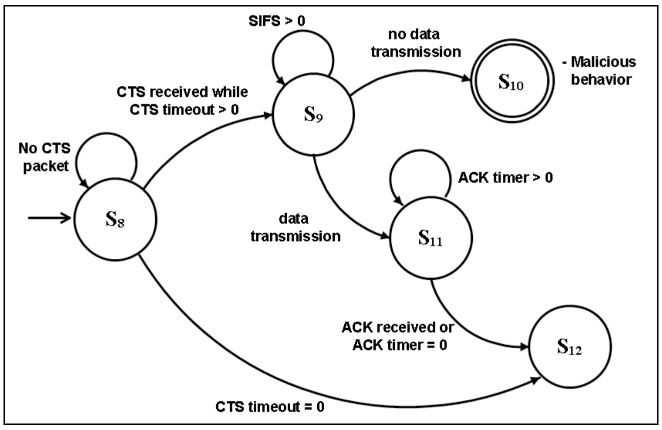

Figure 3. CTS specification diagram.

3) *Backoff specifications*

At state S_{12} the engine monitors for the valid operation of the backoff mechanism. This state can be reached following one of the three scenarios below:

- The communication channel is busy and thus, the transmitting node cannot send its data.
- Either the CTS or ACK timer expired before the corresponding packets arrived at the destination. In this scenario, a collision at the receiver is likely to have occurred.
- The node has successfully transmitted a packet and it must now backoff before transmitting a new one.

The engine retrieves CWmin from the physical layer and a random backoff is generated by the protocol. After that, in the first and third scenario the engine moves to S_{13}, while in the second scenario moves to S_{14}. In S_{13}, if the random backoff is not within the range of [0, CWmin], then the engine moves to S_{15}, which designates a malicious behavior. On the other hand, if the value of the backoff timer is within the expected range, the engine moves to S_{16} and awaits until the backoff plus the DIFS timers expire (the backoff timer is decreased as long as the channel is idle). An attempt by the node to transmit beforehand leads to the final state S_{15}. After the expiration of the timers, if the channel is still busy, the engine returns to S_{12}. Otherwise, it returns to S_6 and the node transmits its RTS packet. Figure 3 illustrates the operation of the engine during this backoff scenario.

Similarly, in S_{14}, if the random backoff is not within the range of [0, 2xCW], then the engine moves to S_{15}, which designates a malicious behavior. On the other hand, if the value of the backoff timer is within the expected range, the engine moves to S_{17} and awaits until the backoff plus the EIFS timers expire. Any attempt by the node to transmit beforehand will lead to the final state S_{15}. After the expiration of the timers, if the channel is still busy, the engine returns to S_{12}. Otherwise, it returns to S_6 and the node transmits its RTS packet.

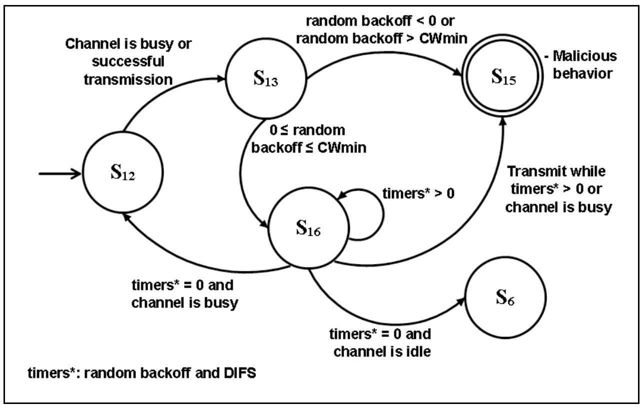
random backoff < 0 or random backoff > CWmin
Channel is busy or successful transmission
S13
S15
- Malicious behavior
0 ≤ random backoff ≤ CWmin
S12
Transmit while timers* > 0 or channel is busy
timers* > 0
S16
timers* = 0 and channel is busy
S6
timers* = 0 and channel is idle
timers*: random backoff and DIFS

Figure 4. A backoff specification diagram.

C. *Reciever specifications*

These specifications exemplify the operation of the 802.11 MAC protocol when the monitored node attempts to receive data from another node. As we have seen in section 3.1, if the monitored node receives an RTS packet, the engine moves to S_2. At this state, the engine retrieves the SIFS parameter from the physical layer and remains at S_2 until the SIFS timer expires. If the node attempts to transmit a CTS before the expiration of SIFS, the engine reaches the final state S_{18}, which designates a malicious behavior. Otherwise, it moves to S_{19}. At this state, the expected behavior of the node is to transmit a CTS packet. If the node transmits a CTS, then the engine moves to S_{20}; otherwise, it moves to the final state S_{18} designating a malicious behavior. At S_{20}, the monitored node is waiting for the actual data packets and thus, the engine will remain in this state until the data timeout timer expires or data are received. In this state the following scenarios are possible:

- No data are received and the data timeout timer expires. In this case the engine returns to the initial state S_0.
- The data timeout timer is expired before it reaches zero. The engine moves to the final state S_{18} designating a malicious behavior.
- Data are received before the data timeout timer expires. The engine moves to S_{21}.

In state S_{21} the engine once again retrieves the SIFS parameter from the physical layer and remains at S_{21} until the SIFS timer expires. If the node attempts to transmit an ACK before the expiration of SIFS, the engine reaches the final state S_{18} which designates a malicious behavior. Otherwise, it moves to S_{22}. At this state, the expected behavior of the node is to transmit an ACK packet. If the node transmits an ACK, then the transmission is completed successfully and the engine returns to the initial state S_0; otherwise it moves to the final state S_{18} designating a malicious behavior.

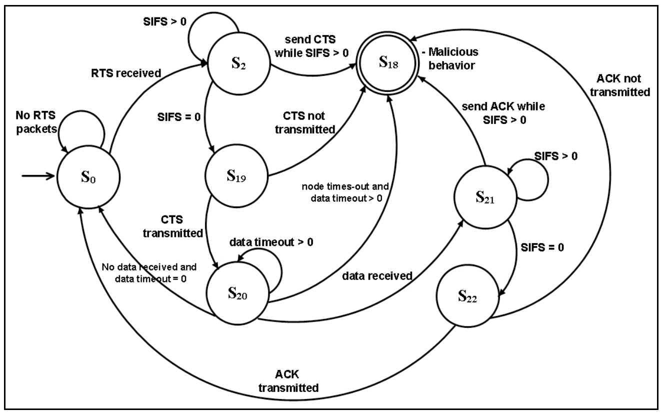

Figure 5. Receiver specification diagram.

IV. Evaluation of the Proposed Detection Engine and Future Work

In this section the proposed detection engine is comparatively evaluated with existing engines for MANETs, in order to ascertain its advantages and limitations. The advantages can help illustrate the contribution of the proposed detection engine, while the limitations can drive future enhancements and optimizations.

A. Advantages of the proposed detection engine

The proposed engine introduces a number of significant advantages over existing detection engines for MANETs, which are analyzed below. First off, the proposed detection engine resolves malicious behaviors in real time. As we have seen in section 3, every protocol operation is monitored by the engine and any activity that results in malicious behavior leads to an immediate breach of the specifications. This advantage is very important since it minimizes the time in which a malicious node can induce damage onto the network. Anomaly-based intrusion detection engines typically resolve attacks in non-real time, since they have to collect audit data for some predetermined time frame, preprocess them, run the detection algorithm and then resolve if a malicious activity takes place [15]. As a result, the detection of an attack takes at least:

$$TF + P + D \tag{1}$$

TF is the time frame, *P* the preprocessing time, and *D* the time it takes for the engine to analyze the audit data.

The proposed detection engine can effectively detect all of the attacks (both known and unknown) that target the operation of the 802.11 MAC protocol. This is achieved by relying on operational constrains (established by the specifications) rather than either focusing on particular attacks (i.e., signature-based detection) or models that statistically characterize the protocols' behavior (i.e., anomaly-based detection). These operational constrains accurately express the expected protocol behavior and thus, any activity that does not act in accordance with these constrains is detected. Signature-based detection engines monitor for predefined patterns of attacks and thus, are unable to detect unknown attacks. On the other hand, in anomaly-based detection, attacks that do not generate a statistical variation greater than some predefined threshold are not detected (i.e., false negatives).

The proposed detection engine is not prone to high rates of false positives, in cases that dynamic changes occur in the network (i.e., churn, changes in the topology, high node's mobility, etc.). These dynamic changes can typically cause detection engines to rely on outdated information and thus falsely consider legitimate behaviors as malicious. Anomaly-based detection engines are prone to such a limitation [15], due to their reliance on a normal profile; which, under dynamic network conditions can become outdated. On the other hand, the proposed engine is not affected by such network conditions because node activities are monitored in real time.

The proposed engine induces limited computational overhead, since the computational complexity of the proposed engine is linear, compared to the state-of-the-art anomaly-based detection engines, which induce polynomial-time complexity [15]. In addition, the induced overhead is uniformly distributed among all the network nodes and thus, there is no unfair distribution of detection responsibilities among the network nodes. The utilization of a host-based architecture also alleviates the need for either audit data exchange or packet monitoring. This advantage not only eliminates any communication overheads, but also a number of security weaknesses, since the exchanged or monitored information might be captured, modified, and retransmitted by a malicious node, in order to mislead the detection engine. Most state-of-the-art detection engines for MANETs (including existing specification-based detection engines) rely on either packet monitoring or audit data exchange and are thus prone to these limitations and weaknesses.

Implementing the engine at the driver level offers two additional advantages: (a) the engine has direct access to both the MAC protocol and the underlying hardware eliminating the existence of any intermediate module or process that may tamper the monitored data; and (b) the detection engine operates as a privileged process (i.e., running on a kernel mode) and thus it cannot be easily circumvented by a malicious process. Furthermore, the implementation does not require any protocol modifications, as it happens in [5]. Monitoring is limited on the 802.11 MAC protocol, and thus, the overhead typically associated with the development of specifications is greatly reduced. Finally, the proposed detection does not require any trusted centralized authorities in order to operate, as happens with DOMINO [12].

B. Limitations of the proposed detection engine

The proposed engine has a number of limitations, which are analyzed below. The detection engine can only resolve attacks that target the operation of the 802.11 MAC protocol. Attacks at higher layers such as a routing table poison attack or a syn flood attack cannot be detected. Furthermore, the detection engine at its presented state is prone to false positives when hardware failures occur. In particular, if a hardware failure occurs at the network interface card, it is likely that the detection engine will pick it up as a malicious behavior. Another limitation is the security of the proposed engine itself. Each engine is responsible for monitoring its host node. If the host node is malicious, it might attempt to disable the detection engine. Finally, the host-based architecture of the engine requires the operation of a detection engine in every node on the network, and thus, nodes are encumbered with intrusion detection responsibilities regardless of their amount of resources.

C. Future work

In future work, the specifications of the proposed engine will be further elaborated and extended in order to: (a) enable the detection of all the attacks that target the critical protocols employed at the transport, network, and data-link layers of MANETs, and (b) distinguish attacks and hardware failures. Moreover, the detection engine will be provided with a more sophisticated IDS architecture, in order to alleviate the need of operating a detection engine at each individual network node.

Additional mechanisms can be utilized to strengthen the security of the detection engine itself. For instance, trusted computing techniques [18], which employ special hardware (i.e., Trusted Platform Modules) to enforce policies, can be applied to safeguard against unauthorized modifications of the detection engine and provide remote attestation. Moreover, reputation based mechanisms [17][19][20] can provide an incentive for nodes to operate the detection engine in order to use the network's resources. Similar techniques applied in malware such as polymorphism, obfuscation, and encryption can also be employed to protect the detection engine from removal or disabling [21].

Finally, simulation studies will be performed to measure the validity of the proposed engine's advantages, and compare it to other existing solutions. More specifically, the proposed engine will be evaluated regarding: (a) the provided detection accuracy, (b) the rate of false positives, (c) the resilience to attacks, and (d) the capability of detecting various attacks at multiple layers.

V. CONCLUSIONS

MANETs are susceptible to a variety of attacks that primarily target the protocols of the transport, network, and data-link layers. Currently, a large number of detection engines have been proposed for MANETs; however, the majority of them present a number of limitations and weaknesses. Furthermore, existing specification-based detection engines for MANETs have only focused on monitoring the network layer and thus, are only capable of detecting routing attacks. The proposed detection engine aims to address these limitations and expand the range of attacks detected by specification-based detection engines. Furthermore, it introduces a number of significant advantages since it can effectively detect both known and unknown attacks in real time and without any communication overhead, it is resilient to the dynamic topologies that are common in MANETs, and its deployment requires no protocol modifications.

REFERENCES

[1] C. Xenakis, C. Panos, I. Stavrakakis, "A comparative evaluation of intrusion detection architectures for mobile ad hoc networks," Computers & Security, Volume 30, Issue 1, January 2011.

[2] C.-Y. Tseng. et al., "A specification-based intrusion detection system for AODV," in proceedings Of 1st ACM Workshop on Security of ad hoc and sensor networks, Fairfax, Virginia, USA, 2003.

[3] C. H. Tseng, T. Song, P. Balasubramanyam, C. Ko, K. Levitt, "A Specification-based Intrusion Detection Model for OLSR", in proceedings of the 8th International Symposium, in recent advances in intrusion detection (RAID 2005), Seattle WA, September 7-9, 2005.

[4] T. Song, C. Ko, C. Tseng, P. Balasubramanyam, A. Chaudhary, K. Levitt, "Formal Reasoning about a Specification-based Intrusion Detection for Dynamic Auto-configuration Protocols in Ad hoc Networks," in proceedings of the 3rd international Workshop on Formal Aspects in Security and Trust (FAST 2005), Newcastle UK, 2005.

[5] Y. Huang and W. Lee, "Attack Analysis and Detection for Ad Hoc Routing Protocols," in proceedings of the 7th international symposium in recent advances in intrusion detection (RAID 2004), Sophia Antipolis, France, Sept. 2004.

[6] J., Orset, B., Alcalde, A., Cavalli, "An EFSM-based intrusion detection system for ad hoc networks." In proceedings of the 3rd international symposium on automated technology for verification and analysis, (ATVA 2005),Taipei,Taiwan, 2005.

[7] H.M. Hassan, M. Mahmoud, and S. El-Kassas, "Securing the AODV Protocol using Specification-based Intrusion Detection," in proceedings of the 2nd ACM International Workshop on Quality of Service & Security for Wireless and Mobile Networks, Torremolinos, Spain, 2006.

[8] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," Jul. 2003. IETF RFC 3561.

[9] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," IETF, RFC 3626, Oct. 2003.

[10] R. Droms. “Dynamic Host Configuration Protocol.” RFC 2131, March 1997.

[11] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11 - 2007.

[12] M. Raya, J. Hubaux, I. Aad, “DOMINO: a System to Detect Greedy Behavior in IEEE 802.11 Hotspots,” in proceedings of the 2nd international conference on mobile systems, applications, and services, , Boston, MA, USA, June, 2004.

[13] NDIS Core Functionality. Retrieved September 29, 2011, from http://msdn.microsoft.com/en-us/library/ff564881.aspx

[14] S. Bearnson, “Communication Basics and Open Data-Link Interface Technology” Novell AppNotes, Nov 1992. Retrieved September 29, 2011, from http://support.novell.com/techcenter/articles/ana19921103.html

[15] C. Panos, C. Xenakis, I. Stavrakakis, “An evaluation of anomaly-based intrusion detection engines for mobile ad hoc networks,” in proceedings of the 8th International Conference on Trust Privacy and Security in Digital Business (TrustBus 2011), Toulouse, August 2011.

[16] B. Wu, J. Chen, J. Wu, and M. Cardei, “A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks” in “Wireless Network Security”, Y. Xiao, X. Shen, and D. -Z. Du , Springer, Network Theory and Applications, Vol. 17, 2006.

[17] S. Zhong, Yang Richard Yang, J. Chen; Sprite: “A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-hoc Networks,” in proceedings of I NFOCOM 2003, pp. 1987-1997, March 2003.

[18] Gang Xu; Borcea, C.; Iftode, L.; , “A Policy Enforcing Mechanism for Trusted Ad Hoc Networks,” Dependable and Secure Computing, IEEE Transactions on , vol.8, no.3, pp.321-336, May-June 2011.

[19] P. Michiardi, R. Molva, “CORE: a Collaborative Reputation mechanism to enforce node cooperation in mobile ad hoc networks,” in proceedings of the Communication and Multimedia Security 2002 Conference, September 2002.

[20] S. Buchegger and J.-Y. Le Boudec, “A Robust Reputation System for Mobile Ad-hoc Networks,” In proceedings of the P2PEcon, June 1-11, 2003.

[21] Alsagoff, S., “Malware self-protection mechanism,” in proceedings of ITSim, Kuala Lumpur, 2008.