Experiences in Passively Detecting Session Hijacking Attacks in IEEE 802.11 Networks

Rupinder Gill **Jason Smith** **Andrew Clark**

Information Security Institute
Queensland University of Technology
GPO Box 2434, Brisbane, Queensland 4001, Australia.
Email: {`rs.gill, j4.smith, a.clark`}`@qut.edu.au`

Abstract

Current IEEE 802.11 wireless networks are vulnerable to session hijacking attacks as the existing standards fail to address the lack of authentication of management frames and network card addresses, and rely on loosely coupled state machines. Even the new WLAN security standard - IEEE 802.11i does not address these issues. In our previous work, we proposed two new techniques for improving detection of session hijacking attacks that are passive, computationally inexpensive, reliable, and have minimal impact on network performance. These techniques utilise unspoofable characteristics from the MAC protocol and the physical layer to enhance confidence in the intrusion detection process. This paper extends our earlier work and explores usability, robustness and accuracy of these intrusion detection techniques by applying them to eight distinct test scenarios. A correlation engine has also been introduced to maintain the false positives and false negatives at a manageable level. We also explore the process of selecting optimum thresholds for both detection techniques. For the purposes of our experiments, Snort-Wireless open source wireless intrusion detection system was extended to implement these new techniques and the correlation engine. Absence of any false negatives and low number of false positives in all eight test scenarios successfully demonstrated the effectiveness of the correlation engine and the accuracy of the detection techniques.

Keywords: wireless intrusion detection, session hijacking, received signal strength, round trip time, passive monitoring

1 Introduction

Session hijacking is a common and serious threat to wireless local area network (WLAN) security (Schmoyer, Lim & Owen 2004). This attack exploits deficiencies in the WLAN state machine, namely unauthenticated management frames and the loose coupling of the IEEE 802.11i and IEEE 802.1X state machines (Mishra & Arbaugh 2003), and can be launched using off-the-shelf hardware and software. Session hijacking combines *denial of service* (DoS) and *identity spoofing* attacks. Typically an adversary forces a legitimate mobile station (STA) to terminate its connection to an access point (AP) by sending it a *disassociation/deauthentication* management frame with the source MAC address spoofed to be that of the AP. This results in the STA disconnecting from the network. The adversary can now associate with the AP, by masquerading the MAC address of the STA, and hence taking over its session. Neither the original IEEE 802.11 standards, nor the recent IEEE 802.11i standard specify mechanisms for protecting the integrity of the management frames, leaving IEEE 802.11 based WLANs vulnerable to management frame spoofing and the associated denial of service attacks that such spoofing permits (Bellardo & Savage 2003). In this paper the terms *Wireless* and *Wireless Local Area Networks* refer to IEEE 802.11 infrastructure networks (IEEE 1999).

This paper extends our earlier work (see Section 2) on detecting session hijacking attacks in WLANs (Gill, Smith, Looi & Clark 2005). In our earlier paper, two new detection techniques were introduced and some basic supporting experiments were conducted to provide confidence in these techniques. However, the experiments concentrated on establishing that the techniques could maintain a manageable false positive rate in the absence of an attacker. Effects of alert correlation and fine tuning of detection thresholds (see Section 6.3) were not explored at all. The contribution of this paper is that it addresses these outstanding issues and demonstrates the accuracy and utility of the intrusion detection techniques proposed in our earlier work through empirical data and use of automated intrusion detection and correlation techniques. To facilitate repeatability and automation, the detection techniques and correlation engine have been incorporated in - *Snort-Wireless*[1], an open source wireless intrusion detection system (WIDS). The experimental scenarios have been designed to study the effectiveness of the detection techniques in the presence of an attacker, who launches an active session hijacking attack on the legitimate STA. The motivation for the scenarios was derived from everyday corporate office environment where fixed and mobile wireless stations coexist. We demonstrate the effectiveness of the correlation engine in dramatically reducing the number of false positives and false negatives. Each detection technique relies on a threshold value which determines which measurement values are treated as anomalous (see Section 6.3). This paper also investigates the effects of fine tuning these thresholds on the number of false positives and false negatives generated by each technique. Measures have also been undertaken to improve the robustness and reliability of the proposed detection techniques. For example, unlike our earlier work, where the timestamp (time when the packet was received by the wireless card) information was derived from *libpcap*[2], the timestamp information is now derived directly from the wireless network card hardware (see Section 3.1).

[1]http://snort-wireless.org
[2]http://www.tcpdump.org

To provide a reasonable degree of realism, these experiments were carried out using real WLAN equipment, with real network drivers and software. The detection techniques were implemented as plugins in Snort-Wireless. However, as these experiments are focused on detailed exploration and analysis, in favour of flexibility and repeatability, real time detection was avoided and the Snort-Wireless plugins were used only to detect attacks from traffic captures. This allowed repeated execution of the IDS plugins over the same traffic captures and facilitated tweaking of the plugin input parameters (threshold values) to study the effects of these changes on intrusion detection results (i.e. the number of false positives and false negatives).

The remainder of the paper is set out as follows. The next section presents an overview of the existing WLAN session hijacking detection techniques and reviews the work presented in our previous paper (Gill et al. 2005). Section 3 discusses the hardware and software configuration for the experiments. The individual experiment scenarios are discussed in Sections 4 and 5. Section 6 presents an analysis of the effectiveness of the detection techniques and the correlation engine in detecting session hijacking attacks. This section also discusses the process followed to discover optimum threshold values for both detection techniques. Finally conclusions and future directions for research are presented in Section 7.

2 Related Work

In our previous paper (Gill et al. 2005), we identified that existing approaches for detecting session hijacking attacks such as monitoring MAC frame sequence numbers and MAC address authentication against precompiled lists can easily be defeated. Ideally a WIDS should utilise unspoofable characteristics from the MAC protocol and the physical layer to enhance confidence in the intrusion detection process. These characteristics should be computationally inexpensive to calculate, allowing them to be determined in a fast and efficient manner. The WIDS should also operate in a passive fashion and not require modification to the standards, wireless card drivers, or client operating system or software. The WIDS should operate, without causing any interference to the live traffic or network performance. As with all IDSs, a WIDS should maintain minimum level of false positives and negatives. Based on these requirements, two new intrusion detection techniques were proposed to detect session hijacking attacks in WLANs: Monitoring Received Signal Strength (RSS technique) and Monitoring Round Trip Times of the Request to Send - Clear to Send (RTS-CTS) Handshake (RTT technique).

Received signal strength (RSS) is a measure of the energy observed by the physical layer at the antenna of a receiver. Radio Frequency (RF) signal strength does not fade in a linear manner, rather it attenuates roughly inversely as the square of the distance between the two communicating nodes (Bardwell 2002). From an intrusion detection perspective, this property is valuable as it is unspoofable and computationally inexpensive to measure. As it is calculated at the receiver, it is also secure from eavesdropping. It is extremely difficult for an adversary to accurately guess the RSS for a sender as perceived by a receiver. The adversary will need to be at exactly the same location as the receiver, use exactly the same radio equipment, and receive the radio signal with same level of interference, reflections and refractions to know the precise RSS value as perceived by the receiver. In the RSS intrusion detection technique, we proposed periodically monitoring the RSS values for a particular STA or an AP from a passive sensor and developing a dynamic profile for the communicating nodes based on their RSS values. Any abrupt or unusual changes could be flagged as suspicious activity indicative of a potential session hijacking attack. The RSS profile would be dynamic, in the sense that it would be rebuilt for every session between two nodes and would be constantly updated with newly observed RSS values for each node per session.

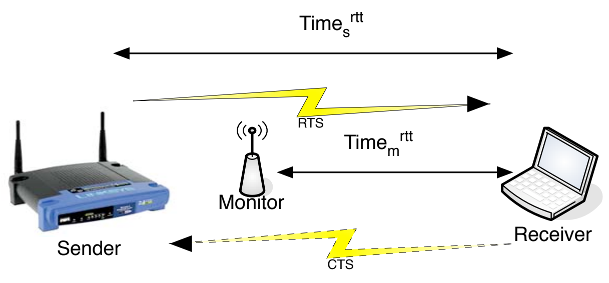

Figure 1: RTS-CTS Round Trip Time (RTT)

In WLANs, before starting transmission, the sender can request positive control over the medium by sending a *Request to Send* (RTS) frame to the receiver. On receipt of the RTS frame, the receiver sends a *Clear to Send* (CTS) frame as an acknowledgment back to the receiver (IEEE 1999). The sender can monitor the time taken for completion of the RTS-CTS handshake between itself and the receiver, $Time_s^{rtt}$. This RTS-CTS handshake between two communicating nodes can also be monitored using a passive sensor. Let $Time_m^{rtt}$ be the time elapsed between when the sensor first detected a RTS frame from the sender to the receiver and when it detected a return CTS from the receiver back to the sender (see Figure 1). This observed RTS-CTS round trip time ($Time_m^{rtt}$) is an unspoofable parameter relative to its measuring entity (i.e. the passive sensor). Its value cannot be guessed by an adversary as it depends on the position of the receiver and the sensor, the distance between the sensor and the receiver and the environment around the receiver and the sensor. It is a measurement relative to the entity measuring it and hence the adversary will have to be at exactly the same location as the sender, using exactly the same radio equipment with same attenuation and antenna gain and receiving the radio waves after same number of reflections and refractions as the sender to accurately predict the values of $Time_m^{rtt}$ between two communicating nodes. It is also protected from eavesdropping as it is a property that is calculated at the passive sensor. In the Round Trip Time (RTT) intrusion detection technique, we proposed that changes in observed time taken for a RTS-CTS handshake to complete between two communicating nodes can be monitored per session by a passive sensor and any abrupt fluctuations can be flagged as suspicious.

In this paper, we extend our previous work by implementing automated IDS plugins to detect anomalies and raise alerts based on RSS and RTT detection techniques. We apply the detection techniques to eight different scenarios to extensively test their usability and accuracy. We study the effects of alert correlation on the number of false positives and false negatives and explore the process of discovering optimum threshold values for the detection techniques.

3 Equipment And Preparation

The experiments were carried out in a lab environment. The same networking hardware/software was used in all experiment scenarios. The following four parties took part in the scenarios: a legitimate client (STA), an access point (AP), a passive Intrusion Detection System (IDS) sensor and an attacker. The AP and the IDS sensor were always stationary in these experiments. However depending on the mobility of the STA, the experiments were divided into two distinct sets. In Set 1, all parties were stationary and in Set 2 the STA was in motion. When in motion, the STA traveled at walking speed, moving across walls, doors and other physical obstacles. In all these experiments, both Snort-Wireless plugin thresholds were set to an arbitrary value of 5 (see Section 6.3 for details). Threshold optimisation was later explored in Section 6.3 to minimise the number of false positives and false negatives. Although the detection techniques proposed in our earlier work (see Section 2) can be used to monitor intrusions against both the AP and the STA, these experiments only deal with session hijacking attacks on the STA.

3.1 IDS Sensor

A *Prism2* chipset based IDS sensor was used in all the experiments as it enables access to physical layer header, called *Prism monitoring header* (PMH)[3] in Radio Frequency Monitoring (RFMON) mode. RFMON mode allows the wireless card to passively monitor all WLAN traffic without any active participation in the network. When Prism cards/drivers receive a wireless frame in RFMON mode, they add their own diagnostics header (PMH) to the frame before passing it over to higher layers for processing. The PMH is not a part of the standard IEEE 802.11 frame header, but is generated by the firmware of the receiving card. It is never actually transmitted and is only used by the receiver for diagnostics. This header includes useful information like *host time* (timestamp when the packet was retrieved from card buffer), *MACtime* (timestamp when the packet was received by the card), *data rate* (Rate at which this packet was received in Mbps), RSSI (Received Signal Strength Indication), Signal strength and Noise, Signal quality etc (Yeo, Banerjee & Agrawala 2002). However, most of the PMH measurements (such as signal strength) do not seem to be in standard units, instead they appear to be vendor specific representations of measurements. However they are still useful from the perspective of a comparative study.

3.2 Snort-Wireless Plugins

The Snort-Wireless plugins developed for these experiments use the PMH to detect anomalies in observed RSS or RTT values for a particular MAC address. The *Signal Strength* field in the PMH is used by the RSS plugin to register the observed RSS values for a particular MAC address. The RSS plugin observes RSS for every wireless frame received, keeps a record of the last observed RSS value per MAC address and checks if the absolute difference between the new observed RSS value for that MAC address and the last one is above a pre-determined threshold (which is configurable). If the threshold is exceeded, then it registers an anomaly in the RSS profile for that MAC address. Similarly, the RTT plugin uses the *MACtime* field (64 bits in length with a resolution of one microsecond[4]) in the PMH of the received RTS and CTS frames to calculate the round trip time (RTT) taken to complete the RTS-CTS handshake between the AP and the STA ($Time_m^{rtt}$). When the plugin detects a RTS frame, it stores its *MACtime* field value along with the MAC address of the destination and then waits for a CTS frame back from that MAC address. When the return CTS frame is detected from the same MAC address, the RTT plugin calculates the RTT by subtracting the stored *MACtime* value of the RTS frame from the *MACtime* of the CTS frame. For every MAC address, the RTT plugin also stores the last observed RTT. Hence when a RTS-CTS event occurs, the plugin calculates the RTT for that MAC address and then compares it against the last observed RTT for that MAC address. If the absolute difference between the two values is more than a pre-determined threshold (which is configurable), it registers an anomaly in the RTT profile for that MAC address. The thresholds of the detection techniques are independent of each other and one can be set to a different value from the other.

Since both the RSS and the RTT measurements are derived from the values registered by the networking hardware in its firmware, these measurements are of maximum possible accuracy that the hardware offers. This factor is especially relevant to RTT measurements. The *MACtime* field represents the time the WLAN hardware actually detected the frame and does not depend on the driver or the operating system or the hardware of the computing platform its attached to. Hence the RTT measurements are not corrupted by interrupt delays and operating system processing queue lengths etc.

3.3 Correlation Engine

In favour of more reliable intrusion detection, the RSS and RTT detection techniques were not used in isolation from each other. Rather results from both the techniques were correlated to provide more confidence in the generated alarms. The correlation engine used is event based i.e. if one of the detection techniques detect an anomaly (an alert), the correlation engine activates and waits until it obtains the detection results from the other technique before making a decision on whether to raise an alarm or not. If both the techniques detected the anomaly, then and only then is an alarm raised (See Figure 2 for correlation engine state machine). For instance if the RSS detection technique registers an abrupt spike in RSS value for a particular MAC address, the correlation engine would register this and then wait for the next RTS-CTS event for this MAC address and check the results of the RTT detection technique. If both techniques registered an alert for that MAC address, then an alarm would be raised. All the RSS events, that occur while the correlation engine is waiting for the RTT technique's results, have no affect on the outcome. An alarm is only raised if both techniques register an alert. Similarly if the RTT technique detects that the RTT taken by the RTS-CTS frames for a particular MAC address has suffered a rapid surge, the correlation engine waits for the next RSS event and only raises an alarm if the RSS technique also registers a spike in the RSS value for that MAC address. The RTS-CTS events for that MAC address, while the correlation engine is waiting for an RSS event, are ignored for the purposes of intrusion detection.

In a real world deployment, the Snort-Wireless plugins would be implemented in the passive IDS sensor and the correlation engine would exist in a central

[3]http://www.ecsl.cs.sunysb.edu/cse591/RFMon.pdf

[4]http://lists.shmoo.com/pipermail/hostap/2003-November/004821.html

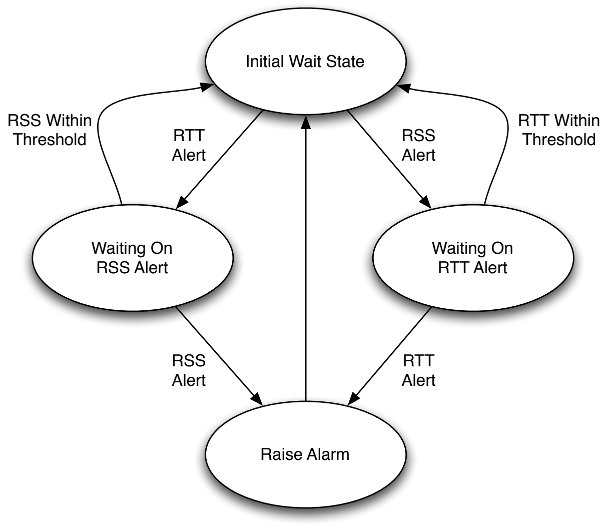

Figure 2: Correlation Engine State Machine

IDS control station. Hence real time attack detection would be possible. In this paper, our aim is to establish the accuracy and usability of the detection techniques and the correlation engine via repeatable and controlled experiments. Therefore, for the purposes of this paper, the passive IDS sensor is only used to create traffic capture dumps. The Snort-Wireless code, implementing the RSS and RTT detection techniques and the correlation engine, is then executed over the offline traffic captures to detect session hijacking attacks.

3.4 Hardware Configuration

A Linksys WRT54g router with *sveasoft*[5] firmware was used as the AP. A laptop with Dlink DWL-650 card, running linux *hostap*[6] driver on Redhat 9 was used as the attacker and a laptop with Orinoco silver card, running orinoco inbuilt driver on Redhat 9 was used as the STA. A Pentium II PC with Netgear MA401 card, running linux hostap driver on Redhat 9 was used in RFMON mode as the IDS sensor (*ethereal*[7] was used for frame capturing). The WRT54g router's *RTSThreshold* option was set to 1 (always on). This enabled RTS-CTS handshake for traffic from AP to STA. No external antennas were used to enhance the reception and no attempt was made to modify the transmission power of any of the wireless equipment.

4 Experimentation - Set 1

In Set 1 of the experiments, the robustness and reliability of the RTT and RSS detection techniques was tested when none of the participants were in motion. The AP, the IDS sensor, the STA and the attacker were all stationary in this set. In all scenarios, the AP was placed in close proximity to the IDS sensor. In Figure 3, points A, B and C represent location of the STA, the AP and the IDS sensor respectively. Points X,Y and Z in Figure 4 represent location of the attacker in Scenarios Two, Three and Four respectively.

[5]http://www.sveasoft.com
[6]http://hostap.epitest.fi
[7]http://www.ethereal.com

4.1 Scenario One

In Scenario One, there was no attacker present and the AP and the STA were placed in close proximity to each other at points B and A respectively (see Figure 3). Network traffic was generated from the STA to the AP. The IDS sniffer was used to capture this WLAN traffic between the STA and the AP. Snort-Wireless, with the RSS and RTT plugins activated, was then run over this traffic capture dump. After examination of 1221 captured frames, Snort-Wireless plugins did not raise any alarms. As there was no attacker present, the detection techniques and the correlation engine correctly did not generate any false positives. An analysis of the Snort-Wireless debug logs also indicated that there were no instances where any one of the two detection techniques disagreed with the other one. Neither of them detected any anomaly at any time.

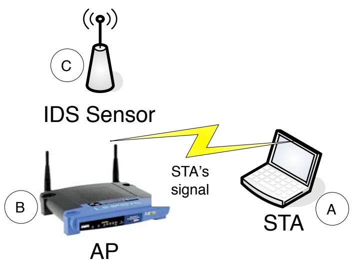

Figure 3: Scenario One

4.2 Scenario Two

In Scenario Two, the AP and the STA were placed in close proximity to each other at points B and A respectively. The attacker was placed in line of sight of the STA at point X (see Figure 4). Then network traffic was generated between the STA and the AP. The attacker then launched a session hijacking attack on the STA.

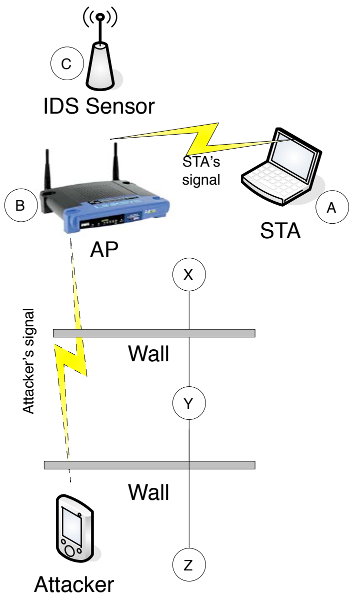

Figure 4: Scenarios Two, Three and Four

Session hijacking attacks are a combination of denial of service (DoS) and identity spoofing attacks. In the real world, a DoS attack would be carried out using a deauthentication attack tool like *void11*[8] and then spoofing attack would involve simply changing the MAC address of the attacker to the STA's MAC address using the *ifconfig* command. However, for the purposes of this paper, to facilitate pinpointing the frame where session hijacking attack started, the DoS attack was simulated by manually switching off the STA. Then the attacker initiated communication with the AP using its own MAC address. All this traffic was captured by the passive IDS sensor. The spoofing part of the attack was simulated by editing the traffic capture dump before passing it on to Snort-Wireless and changing all instances of the attacker's original MAC address to the STA's MAC address.

An analysis of the traffic capture revealed that the session hijacking attack started at frame number 1191. This was the first frame with original source MAC address of the attacker (*00:05:5d:ee:1f:3a*) in the captured traffic dump. Then using a combination of *ethereal*, *sed*[9] and *text2pcap*[10] tools, the captured traffic dump was edited and all instances of the attacker's MAC address were replaced by the STA's MAC address. Hence simulating the spoofing attack in the captured traffic dump. Therefore, now the captured traffic dump file contained a session hijacking attack starting at frame number 1191. All traffic with STA's source MAC address (*00:02:2d:2b:9f:d9*) after frame 1190 was from the attacker. Now Snort-Wireless (with RSS and RTT plugins activated) was run over the captured traffic dump. Executing modified Snort-Wireless over the edited traffic capture (2167 frames) resulted in two alarms.

4.3 Scenario Three

In Scenario Three, the AP and the STA were placed in close proximity to each other at points B and A respectively. The attacker was placed away from of the STA, in a different room, with no line of sight to the STA (at point Y in Figure 4). Then in exactly similar fashion to Scenario Two (see Section 4.2), the remainder of the experiment was conducted. After editing the captured traffic dump and running Snort-Wireless over 2258 captured frames, three alarms were generated.

4.4 Scenario Four

In Scenario Four, the AP and the STA were placed in close proximity to each other at points B and A respectively. The attacker was placed very far away from of the STA (close to the RF range limit of the AP), in a different building, with no line of sight to the STA (at point Z in Figure 4). The remainder of the experiment was conducted in similar fashion to Scenario Two (Section 4.2). After editing the traffic dump and running modified Snort-Wireless over 2005 captured frames, two alarms were registered.

5 Experimentation - Set 2

In Set 2 of the experiments, the robustness and reliability of the RTT and RSS detection techniques was tested with the attacker stationary and the STA in motion between a point closer to the AP and another point far away from it. The AP, the IDS sensor, and the attacker were all stationary at locations B, C and D (see Figures 5 and 6). In all scenarios, the IDS sensor was placed in close proximity to the AP. The equipment setup was exactly as described in Section 3.

[8] http://www.wlsec.net/void11

[9] http://www.gnu.org/software/sed/manual/html_mono/sed.html

[10] http://www.ethereal.com/docs/man-pages/text2pcap.1.html

5.1 Scenario Five

In Scenario Five, the AP and the attacker were stationary and were placed in close proximity to each other (in line of sight at points B and D in Figure 5). Network traffic was then generated from the STA to the AP. The STA then started traveling (at walking pace) from a point close to the AP to a point far away from it (i.e. from point E to F in Figure 5). Towards the end of the STA's journey, the attacker then launched a session hijacking attack on the STA. The steps taken to simulate the session hijacking attack were same as described in Section 4.2. Executing modified Snort-Wireless over the edited traffic capture (1590 frames) resulted in one alarm.

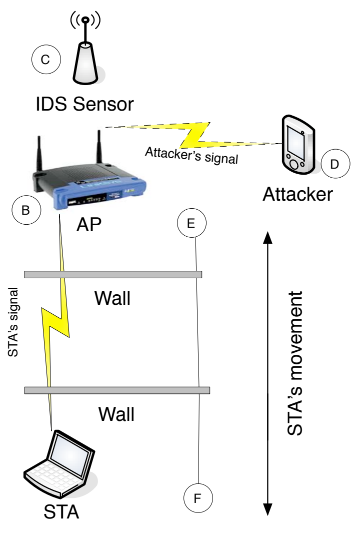

Figure 5: Scenarios Five and Six

5.2 Scenario Six

In Scenario Six, the AP and the attacker were stationary and were placed in close proximity to each other (in line of sight at points B and D in Figure 5). Network traffic was then generated from the STA to the AP. The STA then started traveling (at walking pace) from a point far away from the AP to a point close to it (i.e. from point F to E in Figure 5). Towards the end of the STA's journey, the attacker then launched a session hijacking attack on the STA. The session hijacking attack was simulated in exactly the same fashion as described in Section 4.2. Executing modified Snort-Wireless over the edited traffic capture (1562 frames) resulted in two alarms.

5.3 Scenario Seven

In Scenario Seven, the AP and the attacker were stationary and were placed far way from each other (not in line of sight, at points B and D in Figure 6). Network traffic was then generated from the STA to the AP. The STA then started traveling (at walking pace) from a point at close proximity to the AP to a point far away from it (i.e. from point E to F in Figure 6). Towards the end of the STA's journey, the attacker then launched a session hijacking attack on the STA. The steps taken to simulate the session hijacking attack were same as described in Section 4.2. After editing the capture dump and running modified Snort-Wireless over 1513 captured frames, only one alarm was raised.

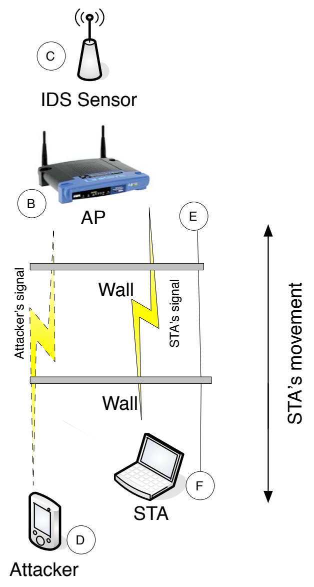

Figure 6: Scenarios Seven and Eight

5.4 Scenario Eight

In Scenario Eight, the AP and the attacker were stationary and were placed far way from each other (not in line of sight, at points B and D in Figure 6). Network traffic was then generated from the STA to the AP. The STA then started traveling (at walking pace) from a point far away from the AP to a point close to it (i.e. from point F to E in Figure 6). Towards the end of the STA's journey, the attacker then launched a session hijacking attack on the STA. The session hijacking attack was simulated in exactly the same fashion as described in Section 4.2. Executing modified Snort-Wireless over the edited traffic capture (3135 frames) resulted in two alarms.

6 Analysis

6.1 True Positives and False Positives

A true positive is the alarm raised when an IDS correctly identifies an abnormal event as an attack, while a false positive is the alarm raised when an IDS misclassifies a normal event as an attack. A false negative is just the opposite of a true positive where the IDS fails to identify the attack and does not raise an alarm. In our experiments, no false negatives were registered i.e. all the attacks were successfully and accurately detected. However some false positives were also raised by the correlation engine.

Tables 1 and 2 summarise the true positives and false positives raised by the correlation engine in all eight scenarios. Column one is the scenario number and column two is the number of true positives/false positives raised in that scenario along with the RSSdiff[11] and RTTdiff[12] values that triggered the alarm. Column three represents the frame numbers (in the captured traffic dumps per scenario), where the RSS and the RTT anomalies occurred. In column three, the frame numbers are represented by a tuple where the first member of the tuple represents the frame number in the captured traffic dump where the RSS alert was registered and the second member is the frame number where the RTT alert was raised. For instance, the entry for Scenario Two in Tables 1 and 2 shows that in Scenario Two, 2 alarms were raised by the correlation engine. The first one at frame 1217 and the second one at frame 1400. The alarm at frame 1217 was caused by a RSS fluctuation (RSSdiff=24) at frame 1191 and a RTT spike at frame 1217 (RTTdiff=147) for the STA. Frame 1217 was the very next RTS-CTS handshake event for the STA after frame 1191. Hence, both the RSS and the RTT plugins reported the anomaly and the session hijacking attack was identified correctly and accurately. However the alarm raised at frame 1400 was a false positive. It was caused by a RSS spike (RSSdiff=20) generated at frame 1397 and a RTT fluctuation at frame 1400 (RTTdiff=132).

Scenario	Number of True Positives	Frame numbers
One	NA	NA
Two	1 (RSSdiff=24) (RTTdiff=147)	(1191 1217)
Three	1 (RSSdiff=37) (RTTdiff=152)	(845 866)
Four	1 (RSSdiff=44) (RTTdiff=161)	(909 912)
Five	1 (RSSdiff=34) (RTTdiff=159)	(842 870)
Six	1 (RSSdiff=16) (RTTdiff=130)	(681 735)
Seven	1 (RSSdiff=22) (RTTdiff=143)	(966 980)
Eight	1 (RSSdiff=36) (RTTdiff=167)	(702 705)

Table 1: True Positives

Scenario	Number of False Positives	Frame numbers
One	NONE	NA
Two	1 (RSSdiff=20) (RTTdiff=132)	(1397 1400)
Three	2 (RSSdiff=15) (RTTdiff=105) and (RSSdiff=18) (RTTdiff=122)	(1067 1400) and (1410 1414)
Four	1 (RSSdiff=13) (RTTdiff=114)	(1099 1106)
Five	NONE	NA
Six	1 (RSSdiff=8) (RTTdiff=64)	(318 332)
Seven	NONE	NA
Eight	1 (RSSdiff=7) (RTTdiff=131)	(283 296)

Table 2: False Positives

In all scenarios, the first alert raised was the RSS alert followed by the RTT alert. This can be explained by the fact that the number of RSS events in a traffic capture are higher than the RTS-CTS events

[11]RSSdiff is the absolute difference between the last observed and the current RSS value for a MAC address

[12]RTTdiff is the absolute difference between the last observed and the current RTT value for a MAC address

and the attacker never starts the session hijacking attack with a RTS-CTS handshake. The attacker will always send a spoofed frame to the AP, leading to a RSS event first. The response transmission from the AP initiates a RTS-CTS handshake, enabling the passive monitor to take the RTT reading for the attacker. Hence the RSS detection technique always detects the session hijacking attack before the RTT technique. Tables 1 and 2 also show that the RTTdiff values in all alarms are higher than the corresponding RSSdiff values. This is most likely a result of the higher resolution of the *MACtime* field as compared to the *Signal Strength* field in the PMH.

An interesting observation was made that in Scenarios Two, Three and Four, where all parties were stationary, all the false positives were detected in frames generated after the attack had commenced. This meant that all the false positives were caused by abnormal fluctuations in observed RSS and RTT values for the attacker. This observation was most likely the result of increasing distance between the attacker and the passive IDS monitor from Scenarios Two to Four. Lack of line of sight connectivity and presence of various obstacles (walls, doors etc.) most likely acted as contributing factors to random fluctuations in observed RSS and RTT values for the attacker. Being positioned in close proximity of the sensor in all these scenarios, the STA did not suffer such random fluctuations and hence did not generate any false positives before the attack was launched.

Scenario	Frame Number the Attack Started at
One	NA
Two	1191
Three	845
Four	909
Five	842
Six	681
Seven	966
Eight	702

Table 3: Frame Number at which the Session Hijacking Attack Commenced

However, in Scenarios Six and Eight, just the opposite was observed. The false positives were detected in frames generated before the attack had commenced, which means that the source of these abnormalities was the STA and not the attacker. In these scenarios, the attacker was always stationary and the STA was in motion. These false positives can be attributed to the fluctuations in observed RSS and RTT values for the STA as a result of it being in motion.

Despite different factors contributing to false alarms in different scenarios, the correlation technique successfully managed to keep the number of these false positives fairly low. No false positives were registered in Scenarios One, Five and Seven and almost all other scenarios registered only one false positive (see Table 2). The detection techniques successfully detected the session hijacking attacks in Scenarios Two to Eight at the precise moment (i.e. frame numbers) when they were launched (see Tables 3 and 1). Hence there were no false negatives.

In Scenarios One to Four, as expected, the RSSdiff and RTTdiff values increased as the attacker was placed further away from the STA. In Scenarios Five and Six, the AP, the IDS sensor and the attacker were located in close proximity of each other and as expected, the RSSdiff and RTTdiff values increased as STA moved away from them and decreased as STA moved closer. In Scenarios Seven and Eight, the attacker was located further away from the IDS sensor and the AP. The observed RTTdiff and RSSdiff values increased as STA moved away from the attacker, and decreased as it moved closer to the attacker (see Table 1).

6.2 Single Anomalies

Analysis of the Snort-Wireless debug logs indicated that there were instances when the two detection techniques disagreed with each other. In this paper, we refer to these disagreements as *single anomalies*. A single anomaly would occur if a RSS alert was registered by the RSS detection technique, however the RTT plugin did not register an anomaly in the next RTS-CTS event for that MAC address. Another example would be if a RTT alert was raised by the RTT plugin but the next RSS reading for that MAC address was under the threshold. A detection technique will only raise an alert if the difference between the last observed and current characteristic is above a threshold. In these experiments, both Snort-Wireless plugin thresholds were set to an arbitrary value of 5 (see Section 6.3 for details). Single anomalies are ignored by the correlation engine and an alarm is only raised if both the detection techniques register an alert.

Table 4 shows the observed single anomalies in each scenario and demonstrates the number of potential false positives per scenario that were successfully avoided by the correlation technique. In Table 4 the single anomaly values have been represented as the tuple (RSSdiff RTTdiff). The single anomaly values in RSS single anomaly column have their RSSdiff values greater than the RSSdiff threshold i.e. 5 (see Section 6.3) and the RTTdiff values lower than or equal to the RTTdiff threshold i.e. 5. Similarly the single anomaly values in RTT single anomaly column have their RTTdiff values greater than the RTTdiff threshold and the RSSdiff values lower than or equal to the RSSdiff threshold. The number of single anomalies is a direct function of the threshold values chosen for each detection technique. As expected, both the RSS and RTT single anomalies increased in number in the last four scenarios, due to the mobility of the STA (see Table 4).

Figure 7 shows the distribution of single anomalies registered by the RSS and the RTT plugins, where the correlation engine did not raise an alarm, even though an anomaly was detected by at least one of the plugins. In Figure 7, Quadrant B represents all readings where the observed RTTdiff is greater than the RTTdiff threshold and the observed RSSdiff is lower than or equal to the RSSdiff threshold. While Quadrant D represents the readings where the observed RSSdiff is greater than the RSSdiff threshold and the RTTdiff is lower than or equal to the RTTdiff threshold.

All RTT single anomalies lie in Quadrant B and all RSS single anomalies in Quadrant D. Table 4 and Figure 7 show that the number of single RSS anomalies per scenario were higher than RTT single anomalies. This demonstrates the usability of the correlation process where the RSS technique, used on its own, would have generated lot more false positives. All single anomalies in each scenario were correctly processed by the correlation engine and silently ignored. This contributed to maintaining a low number of false positives.

6.3 Threshold Optimisation

In all these experiments, thresholds for both the detection techniques were set to an arbitrary value of 5. This means a RSS anomaly was only registered if

Scenario	RSS Single Anomaly Values	RTT Single Anomaly Values
One	NONE	NONE
Two	(9 2), (7 1)	(3 32)
Three	(20 3), (23 2), (28 1), (6 0) (9 4), (13 2), (9 1)	(1 24), (2 7)
Four	NONE	NONE
Five	(8 3), (6 0), (9 1), (7 3), (8 2), (9 4), (11 1), (12 0), (10 3)	(3 51), (3 43), (1 22)
Six	(8 4), (15 1), (9 3), (10 2) (6 1)	(2 17), (1 14), (3 19), (0 15), (2 21), (3 33), (4 24), (4 27)
Seven	(16 4), (6 2), (9 0), (9 3), (12 2), (11 1), (8 0), (12 1), (11 0), (10 2), (13 0)	(1 31), (3 33), (1 24), (0 18), (2 17)
Eight	(11 0), (9 1), (6 0), (9 3), (7 4), (8 4), (17 0), (13 2), (12 0) (8 0)	(2 23), (1 17), (4 21), (3 27), (3 13), (3 26), (1 11), (1 16)

Table 4: Single Anomalies

RSSdiff was greater than 5 and a RTT anomaly was only acknowledged if RTTdiff value was greater than 5. An alarm was raised by the correlation engine only when both the RSSdiff and RTTdiff thresholds were exceeded. Setting the threshold value for each plugin is really just an educated guess followed by empirical fine tuning. The threshold value 5 was thought to be just low enough to avoid a high number of false negatives and just high enough to avoid a large volume of false positives. And since ideally both the techniques should exhibit the same level of accuracy, the same threshold value was used for both. In this experiment both the thresholds were set to the same value, however this was purely coincidental and there is no requirement on the RTTdiff and RSSdiff thresholds to be set to the same value. In the real world, after arbitrarily choosing the thresholds for each technique individually, the number of false positives and false negatives generated would be monitored for that particular deployment. After analysing the generated alarms, fine tuning of the threshold values would be undertaken to maintain lowest number of false positives and false negatives.

From Table 2, it is clear that most of the false positives can be eliminated by fine tuning the RTTdiff and RSSdiff threshold values. Table 5 demonstrates how this tuning of the RTTdiff and RSSdiff threshold values would affect the number of false positives and false negatives. In Table 5, threshold values have been represented as a tuple where first member of the tuple represents the RSSdiff threshold and the second one is the RTTdiff threshold. Starting with relaxed threshold values like RSSdiff threshold of 6 and RTTdiff threshold of 7, we notice a high number of false positives and no false negatives. As the threshold values go higher, the number of false negatives start to increase and the number of false positives start to decrease. If we raise the RSSdiff threshold

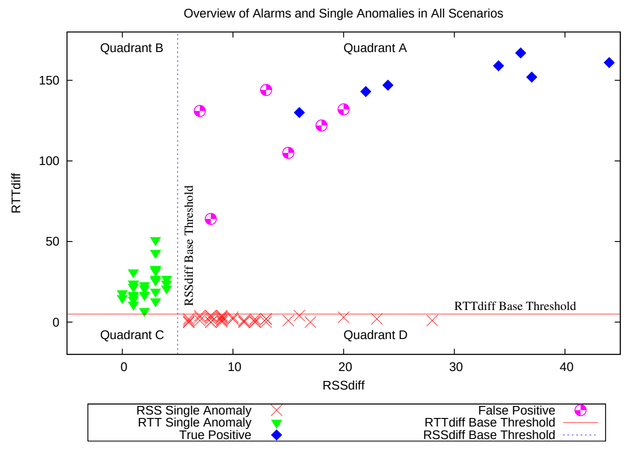

Figure 7: Overview of Alarms and Single Anomalies in All Scenarios

to 20 and RTTdiff threshold to 132, we eliminate all false positives. However, these new threshold values also introduce one false negative. With these threshold values, the true positive in Scenario Six would not get raised (see Table 1). As the thresholds get stricter (higher values), the number of false negatives increase rapidly and the number of false positives are reduced to zero. Higher threshold values tend to lead to more false negatives and lower threshold values result in more false positives (see Table 5). Hence fine tuning of the threshold values has to be a balance between the need for detecting every session hijacking attack (i.e. security sensitivity) and accepted security risk. For single sensor based IDS systems, discovering appropriate threshold values for a particular site can take some time and effort.

Threshold	Number of False Negatives	Number of False Positives
(6 7)	0	6
(7 14)	0	5
(12 24)	0	4
(13 26)	0	3
(15 105)	0	2
(16 28)	1	2
(19 30)	1	1
(20 132)	1	0
(22 140)	2	0
(24 146)	3	0
(28 154)	4	0
(34 159)	5	0
(40 160)	6	0
(50 200)	7	0
(10 170)	7	0
(150 150)	7	0

Table 5: Threshold Fine Tuning

From intrusion detection perspective, it is far more critical for an IDS to minimize the false negative rate than to maintain a low false positive rate. The cost of missing an attack is much higher than the cost of raising a false alarm. Figure 8 shows the ROC[13] curve for the correlation technique used in this paper. It plots the true positive rate of detection against the corresponding false positive rate of error. When relaxed thresholds are employed, the rate of true positives and false positives is high. For instance, at RTTdiff

[13]Receiver Operating Characteristic

threshold of 1 and RSSdiff threshold of 1, the rate of false positives and true positives is 100 %. As the thresholds are raised, the false positive rate starts decreasing. At RSSdiff threshold set to 15 and RTTdiff threshold at 105, the false positive rate drops to lowest 16.67 % while still achieving 100 % true positive rate. If the threshold are raised further, the true positive rate starts decreasing along with the false positive rate and hence leading to false negatives. At RSSdiff threshold of 22 and RTTdiff threshold of 140, the false positive rate hits zero with a true positive rate of 77.4%. Increasing the thresholds to more stricter values introduces more false negatives by decreasing the true positive rate, while the false positive rate stays at zero. Finally at high thresholds like RSSdiff threshold of 50 and RTTdiff threshold of 200, both the true positive rate and false negative rate become zero i.e. IDS becomes unable to detect any attacks with false negative rate of 100 %.

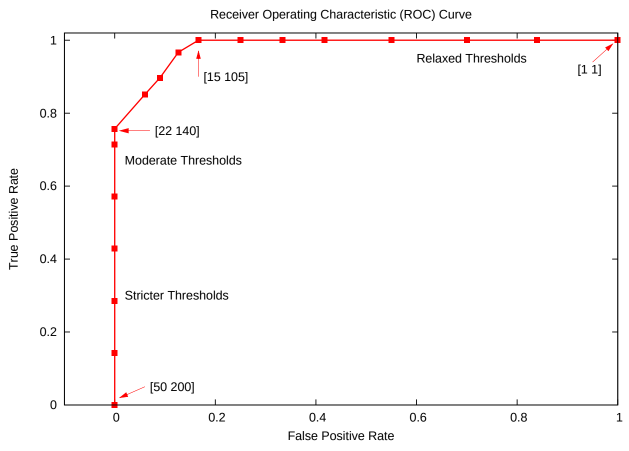

Figure 8: Receiver Operating Characteristic (ROC) Curve

The effects of changes in threshold values on false positives and false negatives have been illustrated in Table 5 and Figure 8. Choosing optimum RSSdiff and RTTdiff threshold values requires analysing the single anomalies, false positives and true positives to select values that minimize the number of false positives and maximize the number of true positives while preventing the single anomalies from becoming false positives. Figure 7 and Figure 9 together present a visual technique to aid in choosing optimum RSSdiff and RTTdiff thresholds.

Figure 7 represents the true positives, false positives and single anomalies registered by the IDS when using the initial threshold settings. In Figure 7, *RSSdiff Base Threshold* and *RTTdiff Base Threshold* refer to initial values used for thresholds (i.e. 5 for all scenarios presented in this paper). Quadrant B represents all readings where the observed RTTdiff is greater than the *RTTdiff Base Threshold* and the observed RSSdiff is lower than or equal to the *RSSdiff Base Threshold*. While Quadrant D represents the readings where the observed RSSdiff is greater than the *RSSdiff Base Threshold* and the RTTdiff is lower than or equal to the *RTTdiff Base Threshold*. Quadrant A represents those readings where both the RTTdiff and RSSdiff values are greater than their thresholds. This quadrant contains all the alarms i.e. true positives and false positives. Lastly Quadrant C represents all normal readings where neither RSSdiff nor RTTdiff reading is above the threshold. Figure 9 represents the true positives, false positives and single anomalies registered by the IDS when using the optimum threshold settings. In Figure 9, *RSSdiff Optimum Threshold* and *RTTdiff Optimum Threshold* refer to fine tuned optimum values of RSSdiff and RTTdiff thresholds respectively. These values are chosen to minimize the number of false positives and false negatives. The Quadrants in Figure 9 are same as in Figure 7, however their areas have changed with the new threshold values.

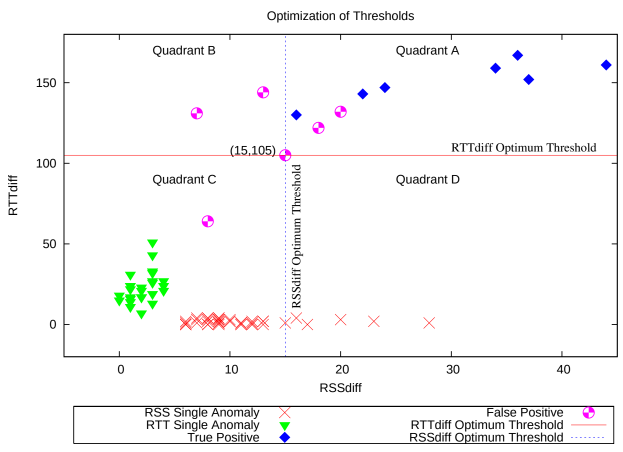

Figure 9: Optimization of Thresholds

Table 5 and the ROC curve (Figure 8) clearly demonstrate that RSSdiff threshold of 15 and RTTdiff threshold of 105 is the optimum choice for the thresholds of the detection techniques. *RSSdiff Optimum Threshold*=5 and *RTTdiff Optimum Threshold*=105 generates no false negatives and raises only two false positives. Figure 9 shows how the single anomalies, false positives and true positives get affected by the new optimum threshold values. As a result of the new thresholds, out of total six false positives, three became RTT single anomalies and moved into Quadrant B and one moved to Quadrant C and got reduced to a normal event. The new thresholds did not introduce any false negatives, that is no true positives were moved out of Quadrant A. None of the single anomalies got moved into Quadrant A, which means that no single anomaly was converted into a false positive as a result of the new threshold. As a result of new thresholds, some of the single anomalies (both RSS and RTT single anomalies) moved to Quadrant C, that is, they became normal events. Hence with 100% true positive detection and four less false positives, *RSSdiff Optimum Threshold* of 5 and *RTTdiff Optimum Threshold* of 105 prove to be the optimum threshold values for the test scenarios presented in this paper. Figure 9, therefore helps in understanding the effects of the new threshold values on single anomalies and the generated alarms (false positives and true positives) and hence assists in choosing optimum thresholds.

The accuracy and efficiency of the IDS depends solely on the choice of suitable threshold values and hence places a large expectation on the threshold values to be optimally tuned. Using a distributed approach, by deploying multiple distributed co-operating IDS sensors, can decrease this expectation on the accuracy of the threshold values. Rather than relying on the alarms generated by a single IDS sensor, the intrusion detection process can be enhanced by using a distributed architecture for IDS sensors. The distributed sensors would report alarms to a central control authority, where the correlation engine would use input from all the sensors to make intrusion detection decisions. The false positives generated can easily be eliminated using the correlation process by simply comparing alarms from different sensors. This also makes it much harder a job for the attacker to launch a session hijacking attack as they will have to guess and spoof RSS and RTT values of the legitimate STA as observed by each sensor. This will require the attacker to be at multiple locations at the same time, hence making it very hard for the attacker to launch an undetected attack.

7 Conclusions And Future Work

Absence of false negatives and a low number of false positives in all our experiments have successfully demonstrated the accuracy, robustness and usability of the RSS and RTT detection techniques and the event based correlation engine (see Section 6). Both the detection techniques accurately and precisely detected the session hijacking attacks at the correct frame numbers and hence proved to be reliable and effective. However, accuracy of the detection techniques depends upon selection of appropriate threshold values. Low threshold values tend to result in a greater volume of false positives while higher threshold values lead to more false negatives. These threshold values vary from site to site and can only be determined through appropriate testing and experimentation. In Section 6.3, we explored the systematic process of discovering the optimum thresholds and also presented a graphical technique to assist in selecting optimum threshold values for the detection techniques based on empirical data.

The event based correlation engine used in this paper proved to be very effective in retaining the number of false positives to a minimum. It only raises an alarm if it registers alerts from both the detection techniques. In absence of the correlation engine, all the single anomalies in Section 6.2 would have been raised as false positives. Therefore, it can be of great assistance in avoiding a storm of false positives when one of the detection techniques has been misconfigured with a very low threshold.

However, by its very nature, the event based model is prone to delay in detection of the attacks. No decision can be made until results from both the RSS and the RTT detection techniques are available. The ultimate goal of intrusion detection process is to improve the correlation process to eliminate all false positives and false negatives and provide fast and accurate attack detection. One alternative is to use a time period based correlation technique, where rather than waiting for the next RSS/RTS-CTS event after an anomaly, the correlation engine makes a decision either way after expiry of a pre-determined time period. When used on its own, it might not be very effective and might lead to many more false positives and negatives. However, if used together, event based and time period based correlation techniques might be able to provide faster and more accurate detection of session hijacking attacks. In this improved technique, if the pre-determined time period has expired and the next event has not happened yet to enable event based correlation engine to make a decision on the alarm, an alarm is raised anyway and later verified by the event based engine when the result of the next event is available. If the next event is not an anomaly then the IDS can *withdraw* its alarm and indicate so in the logs.

Session hijacking attacks are combination of DoS and identity spoofing attacks where the first step involves the attacker spoofing the MAC address of the AP and sending a deauthentication frame to the legitimate STA, forcing it to terminate its connection to the AP. The attacker then spoofs the MAC address of the STA to communicate with the AP. This sequence of events can also be used to eliminate some of the false positives. When an alarm is reported for a MAC address, the correlation engine can check if a RSS anomaly was registered for the AP just before the reported alarm. This anomaly would exist as a result of the attacker spoofing AP's MAC address to send deauthentication frame to the STA.

In our future work, we plan to extend these experiments and study the RSS and RTT intrusion detection techniques in a network with multiple distributed, co-operating IDS sensors. Each sensor will implement both techniques and the alarms from all sensors will get correlated at a central control center. This approach decreases the expectation on the accuracy of the threshold values, as correlation of alarms from a multitude of sources can be used to eliminate the false positives. The IDS sensors will be deployed using easily configurable, lightweight computing platforms like *Soekris*[14] systems, connected back to a single control station. We also plan to trial both the time period based and the hybrid correlation techniques mentioned above. The correlation engine will be enhanced to check anomalies in the AP's RSS profile just before an alarm is reported for a mobile station. This will further help in keeping the false positives to a minimum. We will also conduct experimentation in scenarios where both the STA and the attacker are in motion.

References

Bardwell, J. (2002), Converting Signal Strength Percentage to dBm Values, Whitepaper. Available at `http://www.wildpackets.com/elements/whitepaper/Converting_Signal_Strength.pdf`.

Bellardo, J. & Savage, S. (2003), 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions, *in* 'Proceedings of the USENIX Security Symposium. Washington D.C., USA'.

Gill, R., Smith, J., Looi, M. & Clark, A. (2005), Passive Techniques for Detecting Session Hijacking Attacks in IEEE 802.11 Wireless Networks, *in* 'Proceedings of AusCERT Asia Pacific Information Technology Security Conference (AusCERT2005), Referred R&D Stream, Clark A., Kerr, K., and Mohay, G. (Eds), University of Queensland', pp. 26–38. Available at `http://www.isrc.qut.edu.au/events/auscert2005/proceedings/gill05passive.pdf`.

IEEE (1999). IEEE 802.11 Standard. Available at `http://standards.ieee.org/getieee802/download/802.11-1999.pdf`.

Mishra, A. & Arbaugh, W. (2003), An Initial Security Analysis of the IEEE 802.1X Standard, Technical report. Available at `http://citeseer.ist.psu.edu/566520.html`.

Schmoyer, T., Lim, Y. X. & Owen, H. (2004), Wireless intrusion detection and response: a classic study using main-in-the-middle attack, *in* 'Wireless Communications and Networking Conference, WCNC. IEEE, Volume: 2, 21-25 March', pp. 883–888.

Yeo, J., Banerjee, S. & Agrawala, A. (2002), Measuring traffic on the wireless medium: Experience and pitfalls, Technical report. CS-TR 4421, Department of Computer Science, University of Maryland. Available at `http://citeseer.ist.psu.edu/yeo02measuring.html`.

[14]http://www.soekris.com