Information Security and Computer Fraud, 2019, Vol. 7, No. 1, 7-12
Available online at http://pubs.sciepub.com/iscf/7/1/2
Published by Science and Education Publishing
DOI:10.12691/iscf-7-1-2

A Lightweight Rogue Access Point Detection Algorithm for Embedded Internet of Things (IoT) Devices

Justice Owusu Agyemang, Jerry John Kponyo*, Griffith Selorm Klogo

Faculty of Electrical/Computer Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
*Corresponding author: jjkponyo@ieee.org

Received November 11, 2018; Revised January 04, 2019; Accepted January 19, 2019

Abstract The Internet of Things (IoT) is a new paradigm that enables the convergence of smart objects and the internet. This convergence has led to the creation of an intelligent network that connects all things to the internet for the purpose of exchanging information. The direct connection of IoT devices to the internet makes them susceptible to several security threats. Researchers have developed techniques aimed at enhancing security of IoT devices at both network and application layers. In this paper, we present a real-time and lightweight algorithm, based on information theoretic approach, that enables rogue access point detection for embedded IoT devices. This is to ensure that WiFi-enabled IoT devices can intelligently distinguish between legitimate and rogue access points. We evaluated the performance of the algorithm with respect to the detection rate and also CPU utilization efficiency.

Keywords: *IoT, MITM, IDS, DoS, AP*

Cite This Article: Justice Owusu Agyemang, Jerry John Kponyo, and Griffith Selorm Klogo, "A Lightweight Rogue Access Point Detection Algorithm for Embedded Internet of Things (IoT) Devices." *Information Security and Computer Fraud*, vol. 7, no. 1 (2019): 7-12. doi: 10.12691/iscf-7-1-2.

1. Introduction

The Internet of Things (IoT) is one of the largest technological revolutions of computing. It is estimated that 41 billion IoT devices will be connected to the internet in 2020 with an $8.9 trillion market [1]. This new paradigm is recognized as one of the most important actors in the Information and Communication Technology (ICT) industry for the next years [2].

The application of IoT systems in domains such as public safety, home automation and health care has significant benefits [3]. However, the convergence of these real-world objects and the internet exposes end users to several security threats. These threats can be categorized into the following: cloning of things, malicious substitution of things, firmware replacement, extraction of security parameters, eavesdropping, Man-In-The-Middle (MITM) attack, routing attack and Denial of Service (DoS) attack [4]. Current research areas are focused on enhancing IoT security; providing data confidentiality and authentication, access control within the IoT network, privacy and trust among users and things, and the enforcement of security and privacy policies. Traditional security countermeasures, and privacy enforcement cannot be directly applied to IoT technologies due to three fundamental aspects: the limited computing power of IoT components, the high number of interconnected devices, and sharing of data among objects and users [5].

Researchers have demonstrated how susceptible IoT devices are to attacks [6]. The authors examined the network activity of three IoT devices (the Philips Hue lightbulb, the Belkin WeMo power switch, and the Nest smoke alarm), and demonstrated the ease at which the security and privacy of these devices can be compromised. In the domain of WiFi-enabled IoT devices, [7] demonstrated how IoT devices are susceptible to Man-In-The-Middle (MITM) attack. In this type of attack, an intruder creates a Software-enabled Access Point (SoftAP), also known as a rogue access point, to lure these IoT devices to associate with this unauthorized access point so as to sniff transmitted data (as shown in Figure 1). The authors further described how difficult it is for less feature-rich Operating Systems (OSs) of many IoT devices to differentiate between the legitimate access point and the rogue access point, hence making these devices vulnerable to this kind of attack.

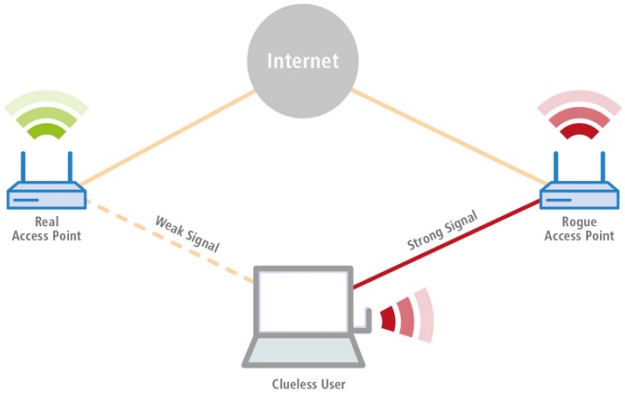

Figure 1. MITM Attack through the use of Rogue AP

The development of IoT products must have the protection of interaction between IoT entities as a concern in order to improve the security of IoT systems. However, auxiliary lines of defense like Intrusion Detection Systems (IDSs) are necessary to act as a line of defense against attackers who may attempt to exploit vulnerabilities in IoT devices.

This paper presents a real-time and lightweight rogue access point detection algorithm that can be implemented on resource-constrained WiFi-enabled IoT devices. We present results on the performance of the proposed algorithm using the detection rate and CPU utilization efficiency as key performance indicators. The rest of the paper is organized as follows: Section 2 reviews related literature. Section 3 presents the motivation for the research. Section 4 describes the methodology. Section 5 discusses and presents results from testing the proposed algorithm. Section 6 is the conclusion and recommendation.

2. Related Works

Rogue access point detection is an import aspect in wireless security. It can be considered as an initial phase of wireless intrusion detection.

A rogue device detection system using techniques such as site survey, Media Access Control (MAC) address list checking, noise checking and wireless traffic analysis has been proposed [8]. The author focused on internal rogue device detection such as wireless devices used by employees in a corporate network. The author used client devices to do periodical scanning to detect rogue access point instead of using dedicated scanning devices. His setup consisted of client devices which communicate with an Access Point (AP) and the access point communicates to a central server. Captured wireless network traffic is sent by clients through the AP to the central server. Clients transfer the captured data to the AP via HyperText Transfer Protocol (HTTP) response. The mode of data transmission between the AP and the central server is via eXtensible Markup Language (XML). The central server is responsible for analyzing the received data and upon detecting an intrusion, logs the intrusion to be analyzed by a system administrator. The approach used in this scenario cannot be applied to embedded IoT devices due to resource constraints. Besides, the various client devices used in the periodic scanning are also susceptible to rogue attacks. There is no defense mechanism to protect the client devices used in the periodic scanning of wireless traffic. Furthermore, in situations where client devices are unable to communicate to the central server through the access points, that means rogue access points cannot be detected.

A conventional rogue access point detection system [9] similar to [8] was also proposed. In their approach, wireless access points are manually set to normal operation or sniffing mode. Captured wireless traffic is then stored in a centralized server and analyzed to detect rogue access points. This approach has limitations similar to that of [8]. Furthermore, it does not incorporate autonomy in its mode of operation, due to the option of manually switching wireless devices modes of operation.

Rogue access point detection based on Received Signal Strength (RSS) was proposed [10]. In this they measured the correlated RSS sequences from nearby APs so as to determine whether the sequences are legitimate or fake. This method works in three phases: The first phase involves the collection of all the RSS from nearby APs. In second phase all these collected RSS are normalized and it estimates the missed RSSs, caused by some external factors and then estimated RSSs are normalizes for generalization of variety of wireless environment. In the last phase, it is determined which RSSs are highly correlated, which is based on some empirical threshold value. The highly correlated RSS sequences are considered as fake signals from a single device.

Mehndi et al. [11], proposed an approach which includes the Mac Address, Service Set Identifier (SSID) and signal strength of access point in order to decide whether the access point is rogue or not. In detecting authorized access points, the MAC addresses of all visible access points are matched against a list of authorized access points. Tools like Ettercap [13], Wireshark [14] and Snort [15] are further used for filtering in instances where the MAC address is spoofed.

3. Motivation

Despite the maturity of IDS technology for traditional networks, current solutions are inadequate for IoT systems, because of IoT particular characteristics that affect IDS development [12]. From the related works, it can be inferred that the conventional approach used in detecting rogue access points are not applicable to IoT devices due to resource constraints; hence the need for lightweight detection algorithms.

This paper presents a novel lightweight rogue access point detection algorithm for embedded IoT devices. The performance of the algorithm is evaluated based on the detection rate and the CPU utilization efficiency.

4. Methodology

The rogue AP detection algorithm works by profiling the legitimate access point using the following characteristics:

- Basic Service Set Identifier (BSSID)
- Channel
- Frequency
- Protocol
- Cipher
- Supported bit rate(s)

The entropy of the legitimate access point is computed using equation 1.

$$H(S) = -\sum_{n=0}^{N} p_i \log_2 p_i \quad (1)$$

Each of the characteristics is assigned a probability by applying the principle of indifference (Bayesian non-informative prior). This is shown in Equation 2. N represents the number of characteristics under consideration.

$$p_i = \frac{1}{N}, N > 1. \quad (2)$$

The algorithm identifies a particular AP as rogue or otherwise by computing its entropy (based on the same characteristics) and comparing the computed entropy with that of the legitimate AP.

4.1. Lightweight Rogue Access Point (RAP) Detection Algorithm

The RAP detection algorithm was implemented in two modes. The first mode uses the ***iwlist*** utility found in the Linux operating system [16]. A tool (parser) for enumerating wireless access points was written based on the ***iwlist*** utility [19]. The tool enables scanning of available wireless access points and the parser is able to filter the desired characteristics based on the results obtained from the scan. The second mode uses a monitoring wireless interface in capturing the wireless traffic. The captured data is analyzed in real-time to determine whether a particular AP found is rogue or legitimate.

In Algorithm 1 (shown in Table 1), a scan is initiated using the ***iwlist*** utility. The captured wireless traffic is then analyzed; indicated by the procedure (line 19). If the SSID of the captured traffic matches the SSID of the legitimate AP, the entropy of the detected AP is computed using the procedure on line 11. The decision rule to flag a particular AP as rogue or not is represented as a unit step function (shown in Equation 3) where ***devEnt*** is the entropy of the legitimate AP.

$$\begin{cases} 1, & <devEnt \\ 0, & >=devEnt \end{cases} \quad (3)$$

Table 1. RAP Detection Algorithm based on *iwlist*

```
1:  while True do
2:      if scan(interface) then
3:          analyze(res)
4:      end if
5:  end while
6:
7:  procedure SCAN(interface)
8:      ← res
9:  end procedure
10:
11: procedure ENTROPY(AP)
12:     entropy = 0
13:     for i ← 0, n do
14:         entropy += −p_i * log_2(p_i)
15:     end for
16:     ← entropy
17: end procedure
18:
19: procedure ANALYZE(res)
20:     if res.ssid == ap.ssid then
21:         if entropy(res) != entropy(ap) then
22:             ← rogue
23:         else
24:             ← legitimate
25:         end if
26:     end if
27: end procedure
```

In Algorithm 2 (shown in Table 2), the RAP detection works by scanning and detecting beacon frames that bears the same SSID as the legitimate AP. It decodes the beacon frame and matches the entropy of its characteristics to that of the legitimate AP.

Table 2. RAP Detection Algorithm based on Beacon Frame decoding

```
1:  while True do
2:      if sniff(mon) then
3:          analyze(res)
4:      end if
5:  end while
6:
7:  procedure SNIFF(mon)
8:      ← pkt
9:  end procedure
10:
11: procedure ANALYZE(res)
12:     pkt = sniff(mon)
13:     if DecodeFrame(pkt) then
14:         if DecodeFrame.ssid == ap.ssid then
15:             if ent(frame) != ent(dev.ssid) then
16:                 ← rogue
17:             else
18:                 ← legitimate
19:             end if
20:         end if
21:     end if
22: end procedure
23:
24: procedure DECODEFRAME(pkt)
25:     if pkt.hasLayer(Dot11Beacon) then
26:         ← decodePacket
27:     else
28:         ← None
29:     end if
30: end procedure
31:
```

4.2. Experimental Setup

The experimental setup consisted of an IoT device built using a Raspberry Pi 3, a TP-Link AP and a virtualized RAP (shown in Figure 2).

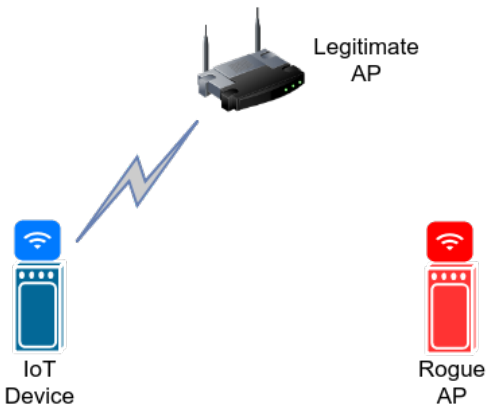

Figure 2. Experimental Setup

4.3. Scenarios

Three scenarios were considered in testing the performance of the RAP detection algorithm. The first scenario implements Algorithm 1 whilst the second and third scenarios implement Algorithm 2 (shown in Figure 3).

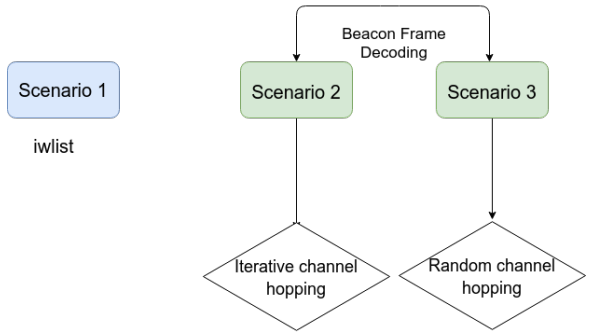

Figure 3. Scenario

Table 3. Channel Hopping Algorithms

```
1:  procedure RANDOMHOP(iface)
2:      n = 1
3:      while True do
4:          changeChannel(n, iface)
5:          temp = int(random.random() * 14)
6:          if temp != 0 and temp != n then
7:              n = temp
8:          end if
9:      end while
10: end procedure
11:
12: procedure ITERATIVE(iface)
13:     while True do
14:         for n ∈ (1, 14) do
15:             changeChannel(n, iface)
16:         end for
17:     end while
18: end procedure
```

In the second and third scenarios, the detection rate is improved by applying a channel hopping technique (Algorithm 3). In scenario 2, a random channel hoping algorithm (Algorithm 3, line 1) was used whereas in scenario 3, an iterative channel hoping algorithm was used (Algorithm 3, line 12). The detection rate and the CPU utilization efficiency was measured while varying the distance between the rogue AP and the legitimate AP.

5. Results and Discussion

The detection rate together with the CPU usage was measured in all scenarios. The distance between the legitimate AP and the RAP was varied from 1 to 30 meters.

The detection rate in Scenario 1 is shown in Figure 4. The detection rate averages **2.15064** seconds (shown in Figure 12).

The CPU usage of Scenario 1 is shown in Figure 5. The CPU usage averages **0.31835%** (Figure 12).

In Scenario 2, the random channel hoping algorithm is applied after the decoding of the beacon frame in RAP detection. The detection rate is shown in Figure 6. The detection rate averages **0.13182** seconds.

The CPU usage in Scenario 2 (shown in Figure 7), averages **2.3049%** (Figure 12).

The Scenario 3 employs iterative channel hopping technique. The detection rate (shown in Figure 8) averages **0.20465** seconds.

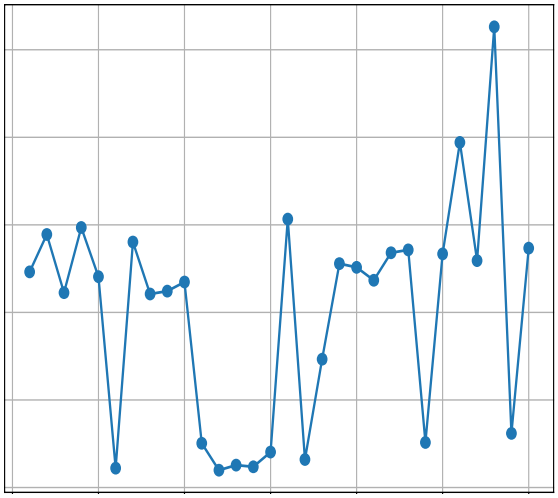

Figure 4. Graph showing the detection rate using Algorithm 1 (Scenario 1)

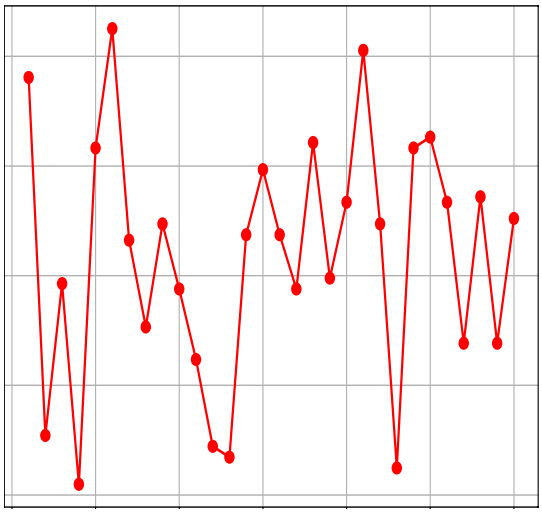

Figure 5. Graph showing the CPU usage using Algorithm 1 (Scenario 1)

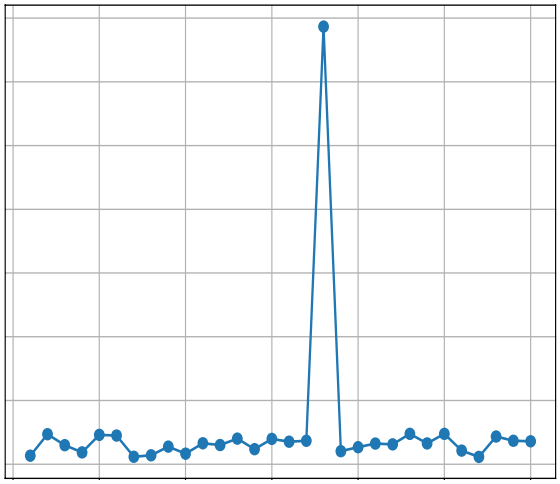

Figure 6. Graph showing the detection rate using Algorithm 2 with random channel hopping (Scenario 2)

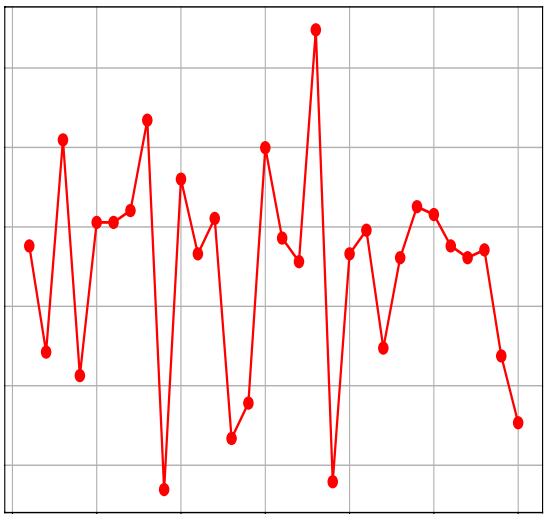

Figure 7. Graph showing the CPU usage using Algorithm 2 with random channel hopping (Scenario 2)

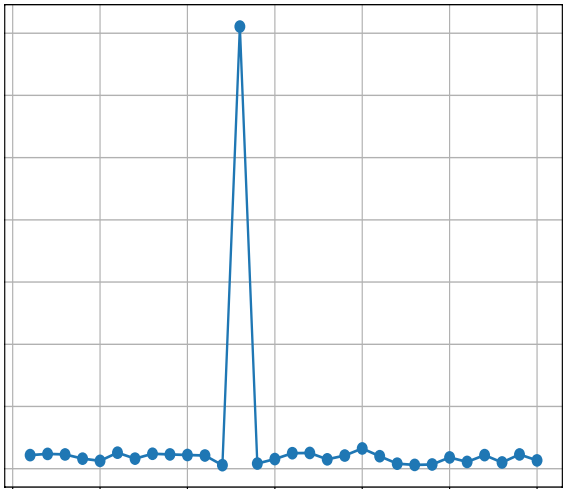

Figure 8. Graph showing the detection rate using Algorithm 2 with iterative channel hopping (Scenario 3)

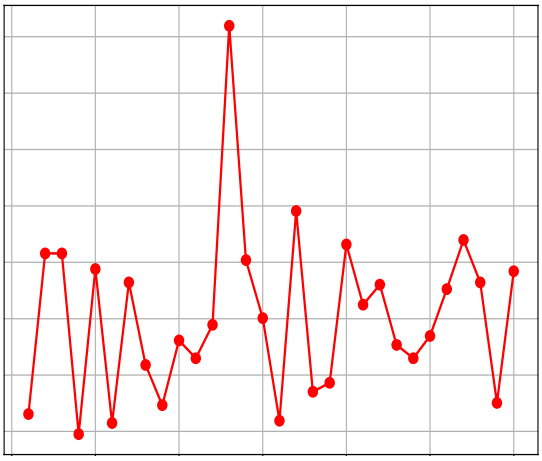

Figure 9. Graph showing the CPU usage using Algorithm 2 with iterative channel hopping (Scenario 3)

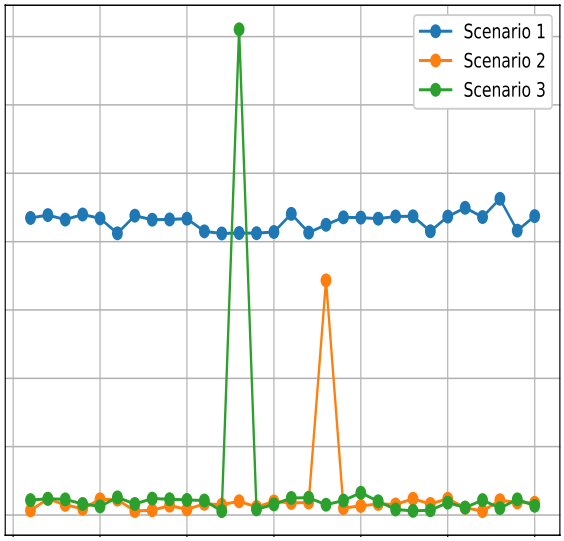

Figure 10. Graph showing the comparison of the detection rate in all three scenarios

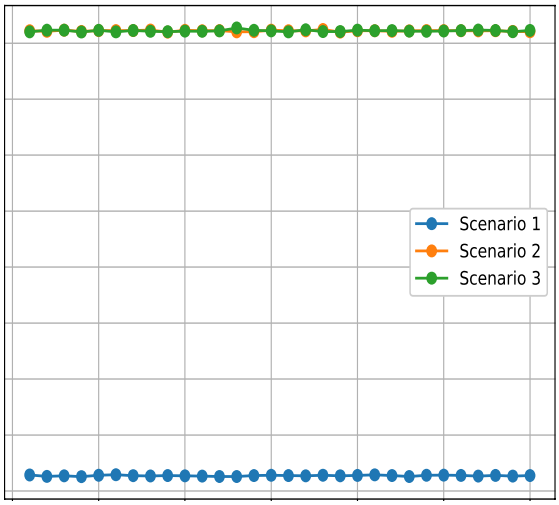

Figure 11. Graph showing the comparison of the CPU usage in all three scenarios

The CPU usage of Scenario 3 (shown in Figure 9) averages **2.3051%**.

Figure 10 shows the comparison of the detection rate of Scenarios 1, 2 and 3 over the measured distance. It can be observed that Scenarios 2 and 3 have better detection rates as compared to Scenario 1.

In Figure 11, Scenario 1 performs better with respect to CPU utilization as compared to Scenarios 2 and 3.

In Figure 12, Scenario 2 and 3 outperform Scenario 1 with respect to the detection rate. On the other hand, Scenario 1 outperforms Scenarios 2 and 3 in terms of efficient CPU utilization.

From the above results, Algorithm 1 can be implemented on embedded IoT devices that do not transmit data at very short intervals i.e. ***intervals > 2.5*** seconds. For embedded IoT devices that transmit data at very short intervals, Algorithm 2 can be implemented since a CPU usage of **2.3%** is quite efficient.

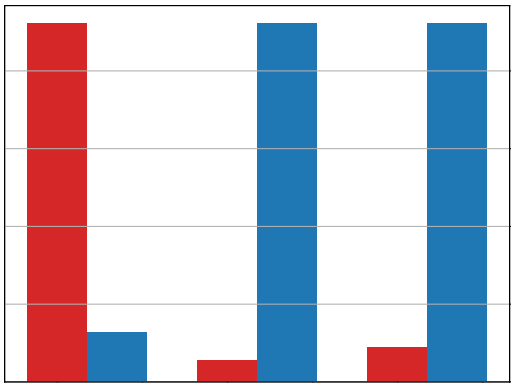

Figure 12. Comparison of the average detection rate and CPU usage of all three scenarios

6. Conclusion and Recommendation

With the emergence of IoT and its security challenges, this RAP detection algorithm will help provide a layer of defense (an IDS) for WiFi-enabled embedded IoT devices. This algorithm addresses only one aspect of the security issues in embedded IoT devices. Future works will explore lightweight detection algorithms that address MITM attacks and also develop an orchestration framework that can be used to logically isolate of IoT devices in instances where they have been compromised. The entire source code for the research can be found at [20].

References

[1] IoT Analytics. (2014). "Why the Internet of Things Is Called Internet of Things: Definition, History, Disambiguation", [Online]. Available: *https://iot-analytics.com/Internet-of-things-definition/*.

[2] Miorandi D., Sicari S., De Pellegrini F., Chalmatac I., "Internet of Things: vision, applications and research challenges", *Ad Hoc Network* 10(7), 1497-1516, 2012.

[3] Borgia, E., "The Internet of Things vision: key features, applications and open issues", Comput. Commun. 54, 1-31, 2014.

[4] Garcia-Morchon O., Kumar S., Struik R., Keoh S., Hummen R., "Security considerations in the IP-based Internet of Things", IETF Internet-Draft, 2013.

[5] Sicari S., Rizzardi A., Grieco L., Coen-Porisini A., "Security, privacy and trust in Internet of Things: the road ahead", Comput. Netw. 76 (0), 146-164, 2015.

[6] Notra S., Siddiqi M., Gharakheili H., Sivaraman V., Boreli R., "An experimental study of security and privacy risks with emerging household appliances", Communications and Network Security (CNS), 2014 IEEE Conference on, pp. 79-84, 2014.

[7] Kolias, C., Stavrou, A., Voas, J., Bojanova, I., Kuhn, R., "Learning Internet-of-things security 'Hands-on'". IEEE Secur. Priv. 20 (February), 2-11.

[8] Ibrahim Halil Saruhan, "Detecting and Preventing Rogue Devices on the Network", SANS Institute, pp. 5-7 2007.

[9] S.B.Vanjale, Amol K. Kadam, Pramod A. Jadhav, "Detecting\& Eliminating Rogue Access Point in IEEE 802.11 WLAN", International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN) Volume-1, Issue-1, 2011.

[10] T. Kim, H. Park, H. Jung, H. Lee, "Online detection of fake access points using received signal strengths", 2012.

[11] Mehndi Samra, Mehak Mengi, Sparsh Sharma, Naveen Kumar Gondhi, "Detection and Mitigation of Rogue Access Point", Journal of Scientific and Technical Advancements, Volume 1, Issue 3, pp. 195-198, 2015.

[12] Bruno Bogaz Zarpelao, Rodrigo Sanches Miani, Caludio Toshio Kawakani, Sean Carlisto de Alvarenga, "A Survey of Intrusion Detection in Internet of Things", Journal of Network and Computer Applications, pp. 2-4, 2017.

[13] Ettercap, http://ettercap.github.io/ettercap/, [Accessed Dec 11, 2018].

[14] Wireshark, http://www.wireshark.org/, [Accessed Dec 11, 2018].

[15] Snort, http://www.snort.org/, [Accessed Dec 11, 2018].

[16] Iwlist, https://linux.die.net/man/8/iwlist, [Accessed Dec 11, 2018].

[17] RaspberryPi, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, [Accessed Dec 11, 2018].

[18] Tenda Wireless Driver, https://github.com/Mange/rtl8192eu-linux-driver/, [Accessed Dec 11, 2018].

[19] Iwlist Parser, https://github.com/jayluxferro/iwlist-parser, [Accessed Dec 11, 2018].

[20] IoT-IDS, https://github.com/jayluxferro/IoT-IDS/.