CSEC Alarm, or Another Rogue Access Point Detector (ARAPD)

https://github.com/jonniesweb/CSEC-alarm
Jonathan Simpson
April 10, 2016

1 Introduction

1.1 Context

Wireless networks are insecure. They are insecure because anyone can setup a wireless access point using the same network name of another existing access point and the clients could unknowingly roam from the original access point to the attacker's access point. Attackers can utilize this feature of wireless networks and intercept network traffic [7].

Attackers can use rogue access points to intercept data, gather information and gain further access into an organization. Security researchers have been working with news outlets to inform people and organizations to be wary of open Wi-Fi networks and to practice proper wireless network security because of the possibility for bad actors to intercept their traffic [6][8]. The Payment Card Industry (PCI) has also recognized that secure wireless networks are necessary for any system that deals with credit cards, and has developed a report with security guidelines [9].

This is an issue for organizations who have deployed wireless access points across their property for the convenience of their users. The organization wants to keep their wireless network secure and free of snoopers. IT departments are usually the ones in charge of managing and maintaining the wireless networks. One of the tools that they have at their disposal is a rogue access point detector. This report presents an implementation of one. A rogue access point detector works by passively fingerprinting each access point in the area and checking if they belong to a set of trusted access points. When there are access points that are not part of the trusted access point set then the IT department would be notified.

1.2 Problem Statement

This report aims to address the problem of detecting when a rogue access point appears in a monitored area and alerting people who should take further action. The rogue access point detector should be able to classify untrusted access points from trusted access points and send out an alert when needed. This is a pertinent problem because the vulnerability has not been fixed and the user's or organization's sensitive data can be snooped if a user's wireless device was connected to a rogue access point. The user is not prompted when their device roams from one access point to another, thereby enabling rogue access points to intercept the device's communication without the user knowing.

Security on wireless networks are often traded for usability. Because of this, a proactive organization should work at protecting the user's and the organization's security when it comes to wireless networking by monitoring their premises for rogue access points.

1.3 Result

Discussed in this report is an implementation of a rogue access point detector that passively monitors access points within an area. The beacons that access points send out are picked up by the rogue access point detector are then compared to a trusted list of access points. The detector then decides whether the received beacon is from a trusted access point or a rogue one. If a rogue access point is detected, an alert is issued to an administrator to perform further action. Organizations can benefit from rogue access point detectors because it can alert them when a non-trusted access point starts advertising itself. Network Administrators are provided with a method to detect rogue access points and act on them only when they appear, thereby freeing up the manual work done by a Network Administrator.

The rogue access point detector program is shown to have a low CPU usage compared to similar rogue access point detectors that are available. Future work can make it easier for administrators to configure the list of trusted access points, and extend the fingerprinting capabilities of access points to make it harder for a rogue access point to spoof a trusted access point.

1.4 Outline

The rest of this report is outlined as follows. Section 2 provides background information about the 802.11 protocol, types of rogue access point detectors, and features of other rogue access point detectors. Section 3 describes the implementation of the rogue access point detector in detail. In Section 4, the implemented rogue access point detector is compared to other rogue access point detectors. Lastly, Section 5 ends the report.

2 Background Information

Wireless networks were created to make mobile networking more convenient. Before wireless communication, devices relied on wired ethernet or other communication methods for connecting multiple devices together. Wireless networking allows for devices to communicate with a network using radio frequencies. The standard for this is called 802.11. For 802.11 wireless access points to cover a physical area, multiple access points may be needed. Some factors that can reduce the usable area covered by an access point could be the load placed on an access point by many clients, the access point signal strength, or even the distance between the access point and a client.

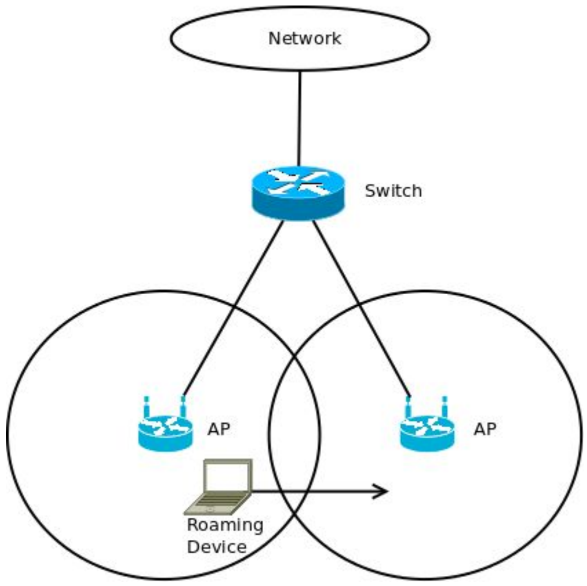

Figure 1: A device roaming from one access point to another

The wireless LAN standard allows for clients to roam from one access point to another as long as the access points have the same ESSID [1]. The ESSID, or Extended Service Set Identification, is a textual representation of the network, also known as the network name. The ESSID is usually human readable since it is the primary means for a user of a wireless client to discover and connect to a wireless network. Roaming allows for wireless clients to move from one access point to another without having to reconnect. The seamless negotiation of roaming to a new access point is simple because access points only need to be configured with the same ESSID. Clients assume that both access points belong to the same physical network when the ESSIDs are the same. Figure 1 shows a device roaming from one access point to another access point on the same network. The device should continue to have access to the network after roaming from one access point to another.

Large corporations and educational institutions that provide a wireless network configure and operate their network to allow for wireless clients to roam from one access point to another. The IT employees are tasked with managing, maintaining and keeping the wireless network safe for the users.

Since it is trivial to setup an access point and set the ESSID to any value, IT departments and their Network Administrators should monitor their network and the surrounding area for any access points that are not theirs. Attackers, or even other employees of the company could set up access points configured with the same ESSID of the organization’s which could pose as a security threat. Sensitive information could be intercepted between the wireless client and the network, or the private network could be infiltrated by these rogue access points. Figure 2 shows wireless Client 1 connected to a trusted access point and Client 2 who is connected to a rogue access point. The rogue access point can intercept any traffic sent between Client 2 and the network. Client 2 is unaware of this happening because both access points share the same ESSID.

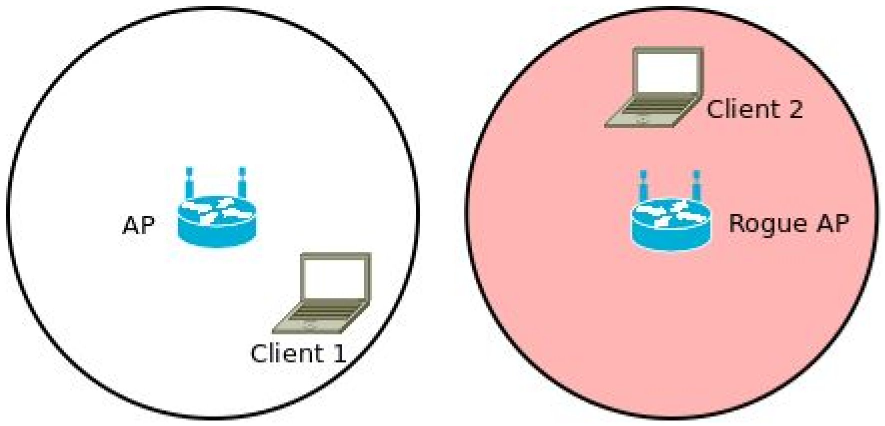

Figure 2: Client 1 is connected to a trusted access point, whereas Client 2 is connected to a rogue access point.

The IT department's goal of keeping the network and its clients secure is made harder by these unauthorized access points. These access points are called "rogue" since they are not under the IT department's control. For IT departments to keep their network secure and free of these rogue access points, the IT department must be proactive in detecting and removing them.

There are two categories of rogue access point detection: one is passive detection, where an agent passively listens to the access points in the area, and the other is active detection, where an agent actively detects and connects to unknown access points. This report focuses on a passive detector.

In both passive and active detectors, a set of access points that are trusted by the IT department are added to the configuration of the detector. Each access point in the set consists of a tuple of data, each value being a characteristic of the access point. Some characteristics that can be read by a detector is the mac address, channel, ESSID, supported rates, encryption parameters and vendor information [3][4]. When the passive or active detector is running, it would compare the access points it detects with the access points in the trusted set. If any access point has been detected but is not in the trusted set then further analysis would start, an alert would be issued, or both.

In a passive detector any data that is sent over a WLAN can be used as a method to classify a trusted access point from an untrusted one. A WLAN is defined by the 802.11 standard. In the OSI model it covers the physical and data link layers. Additionally, the data link layer is split into the Logical Link Control (LLC) and Medium Access Control (MAC) sublayers. By reading the data from these layers it's trivial to determine the access points in the area. Periodically a beacon is broadcasted by every access point to make itself known to other wireless devices in its range. This beacon contains information that can be used to differentiate one access point from another.

The key to determining whether one access point is good or bad relies on being able to fingerprint them. Based off of the data from the beacon frames, timing between beacon frames, and even the data data sent by the access point can be used to create a unique fingerprint. Gathering these fingerprints can either be accomplished by an administrator entering the data manually into the configuration of the rogue access point detector software, or the software itself could enter a “listen” mode where the current access points that it detects are automatically added to the configuration.

There are a few other programs that provide rogue access point detection with varying features and cost. Kismet is an open-source wireless network monitor that has the option to detect and alert for rogue access points. Its trusted access point configuration consists of entering the ESSID and BSSID of each access point into a file that Kismet then reads. Whenever an access point is detected that has the same ESSID but does not have a matching BSSID, an alert is issued. Kismet lacks in the ability to specify more information when fingerprinting access points. Kismet also doesn’t protect the case where a rogue access point uses the same ESSID and BSSID of a trusted access point, but operates on a different channel. The rogue access point will go unnoticed and not cause an alert.

Another rogue access point detector and alerter, EvilAP_Defender, is an open-source project that offers better detection of rogue access points than Kismet. Besides detecting the usual rogue access point with a different ESSID, EvilAP_Defender also takes into consideration the wireless channel, cipher, authentication and even the additional parameters included in beacon frames [5]. This detector tool goes one step further in preventing clients from using those rogue access points by launching a deauth attack on clients that attempt to or are already connected to those wireless networks. The legality of running a deauth attack to protect the clients of your network is illegal in many countries [10].

3 Result

To be able to detect rogue wireless access points, the program must be able to observe the wireless activity on each channel. A small C program was created to interface with the Linux kernel to access and read the 802.11 protocol data units (PDUs) going over the air. This program was nicknamed the packet-monitor because of its behaviour of monitoring the 802.11 PDUs. The packet-monitor would open a raw socket on the wireless interface which has monitor mode enabled. The packet-monitor would then filter for packets that are 802.11 Management packets containing a beacon header. The beacon header contains an important piece of information, the ESSID, because this is the field that wireless clients use to determine which access points they can roam to. Other important information that is extracted from the 802.11 header is the access point’s MAC (BSSID) address. Lastly, the channel is gathered from the beacon. These three values are then outputted to standard out, one entry per line, ready to be read in by another program.

Another program reads in the standard out of the packet-monitor program and performs the main logic required to detect and alert for rogue access points. This program is called the ap-analyser, named after its behaviour of deciding whether an access point is rogue or not. The ap-analyser is configured by the user initially with a set of tuples, where each tuple contains the channel, MAC address of the interface and ESSID of every access point that is trusted by the user. When the ap-analyser starts, it reads in the configuration file and then begins to parse the output of the packet-monitor. Line by line, the ap-analyser compares the current input from the packet-monitor with the tuples found in the configuration. If a match exists, then the input is deemed a trusted access point. If the input matches matches an ESSID from the configuration, but differs in the channel, MAC address, or both, then the access point is deemed to be rogue. When an access point is considered rogue by the ap-analyser an alert is created with information about the rogue access point, and sent to someone who can take further action. In this implementation the alert is appended to an alert log file, but could be configured to send an email or notify another system.

To get the wireless network interface into a proper monitoring state, a shell script was created to automate the process of putting the wireless network interface into monitor mode and change the wireless channel. This script put the wireless network interface into monitor mode using airmon-ng, a wireless monitoring tool, then iterated through channels 1 through 11, each time running the packet-monitor program for a short period of time. Though this script is small, it orchestrates all of the setup and channel switching needed for the packet-monitor to scan all 11 channels.

Both the packet-monitor and ap-analyser programs, as well as the script that coordinates the setup and channel switching for the packet-monitor, a rogue access point detector is constructed. With default settings it takes around 11 seconds to switch through and scan all 11 channels. Therefore the upper bound of detecting when a rogue access point has started advertising in the area takes less than or equal to 11 seconds. Periodically, the program outputs a summary of the number of received beacons. This is here to give the Administrator an idea of how much wireless activity is going on on each channel.

Due to the fact that a single wireless adapter can only listen to one channel at a time means that the packet-monitor program can’t listen to all channels simultaneously. Therefore the packet-monitor takes a best-effort approach of scanning over each channel, listening to each channel for a second. It is common for access points to send out a beacon every 100ms [2]. Therefore listening to each channel for one second allows the packet-monitor to capture multiple beacons from each access point. One consideration to make is that if the beacon interval of an access point was set to more than one second, it would be possible for the packet-monitor to not detect any beacons from that access point for a period of time. Changing the time to listen on each channel to a longer interval than one second increases the chances of hearing that individual beacon, but increases the time it takes to listen to all channels.

For the program to correctly function, the administrator enters a list of trusted access points into a configuration file called *apanalyser.properties* which is located at the base directory of the program. An example configuration is as follows:

```
1,EC:E1:A9:F0:4F:E0,CU-Wireless
1,EC:E1:A9:F0:4F:E1,Pretty Fly For A Wifi
```

The format is the channel, followed by mac address, followed by the ESSID, all separated by commas. The entries in this configuration file are used by the program to determine which access points are trusted and which are rogue. Access points that do not have an ESSID in the configuration file are ignored entirely.

To just run the packet-monitor program, the administrator executes the script *script/run-monitor*. That then starts the packet-monitor program and starts outputting access point information to standard out. Here are the first few lines from the output of packet-monitor:

```
1,00:FC:8D:38:50:58,RogersD2110
1,84:94:8C:98:5D:08,beatrix
1,00:23:6A:72:E2:F3,Primus-e2f1
1,44:E9:DD:4E:22:9A,BELL311
1,BC:4D:FB:73:3B:48,Romulus
```

Each line of the output represents a beacon received from an access point in the detectors proximity. A Ctrl-C or SIGINT sent to the process causes the packet-monitor terminate and free up any resources it has taken.

The packet-monitor program is usually started and stopped by the ap-analyser program. Starting the ap-analyser program involves executing *script/run*. The following is a snippet of what ap-analyser outputs to standard out:

```
Beacons received:
Channel: 1 count: 31
Channel: 2 count: 21
Channel: 5 count: 4
Channel: 6 count: 17
Channel: 9 count: 4
Channel: 10 count: 3
Channel: 11 count: 2

Rogue AP detected! Entry [channel=1, mac=F4:0F:1B:DA:46:D0, ssid=CU-Wireless]
```

Every ten seconds a summary of the number of beacons received is displayed. This shows the current activity of the wireless access points in the area by channel. Below the beacon summary is an alert issued by the ap-analyser. The alert shows that a rogue access point has been detected with details such as the channel, BSSID and ESSID.

Additionally, Administrators might not want to monitor the standard out of the ap-analyser program, therefore the rogue access point alerts also go to a separate log file. The log file is called *alert.log* and is updated with all rogue access point alerts as the occur. A few lines from the *alert.log* file is as follows:

```
[09:13:36 INFO  alert:247]: Rogue AP detected! Entry [channel=1,
mac=F4:0F:1B:DA:46:D0, ssid=CU-Wireless]
[09:13:36 INFO  alert:247]: Rogue AP detected! Entry [channel=1,
mac=F4:0F:1B:DA:42:60, ssid=CU-Wireless]
[09:13:41 INFO  alert:247]: Rogue AP detected! Entry [channel=6,
mac=F4:0F:1B:D8:AE:50, ssid=CU-Wireless]
```

The time of when the alert occurred, followed by information about the alert is what each line of the *alert.log* file contains. The *alert.log* file is designed to be monitored either by another system or an Administrator who would then decide what to do.

4 Evaluation

Kismet, an open source wireless network monitoring tool, has a feature for configuring and detecting when a rogue access point starts broadcasting. The rogue access point detection and alerting offered by Kismet is basic compared to the detector described in this report. Specifically, specifying which channel a known wireless access point runs on is not part of Kismet's configuration, but is a feature of the program in this report.

CPU, memory usage and disk IO will be compared between the rogue access point detector and Kismet. Both the program from this report and Kismet offer rogue access point detection. Because both programs have the same functionality it is possible to compare the two directly. Knowing which program performs better than the other based on CPU, memory usage and disk IO can matter when choosing between rogue access point detectors.

Both Kismet and the rogue access point detector program from this report were run on a laptop computer with a Intel Core i5-2450M quad core CPU at 2.5GHz, 8GB of RAM, Intel Centrino Wireless-N 1000, Ubuntu 14.04.4 LTS, Linux 3.13.0-83-generic. The command line program *pidstat* was used to conduct the performance analysis. The *-T ALL -h -d -u -r* parameters were used and the interval between reporting values was 5 seconds. The *-T ALL* flag was specified to monitor the process and its children processes.

The evaluation environment contained one trusted access point as well as four other wireless access points. A fifth access point was used and configured as a rogue access point. The configuration of Kismet and the rogue access point detector program had the trusted access point's information.

	usr %	sys %	tot %	rss/KB	read KB/s	write KB/s	usr-ms	sys-ms	tot-ms
Average	0.5	0.1	0.5	136947	0	1.6	23	3	26
Median	0.4	0	0.4	136896	0	1.6	20	0	20

Table 1: Performance results of the rogue access point detector

	usr %	sys %	tot %	rss/KB	read KB/s	write KB/s	usr-ms	sys-ms	tot-ms
Average	0.7	0	0.8	3300	0	0	36.5	2.2	38.7
Median	0.8	0	0.8	3300	0	0	40	0	40

Table 2: Performance results of Kismet

The results of the analysis have been summarized in Table 1 and Table 2. Both tables show the average and median of the user time CPU usage (usr %), the system time CPU usage (sys %), the total CPU usage (tot %), the physical memory used by the process (rss/KB), the disk reads (read KB/s), the disk writes (write KB/s), the amount of milliseconds running at user level (usr-ms), the amount of milliseconds running at system level (sys-ms), and the amount of milliseconds running at both user and system level (tot-ms). Pidstat was run over 120 seconds, reporting values every 5 seconds which were then parsed and used to calculate the average and median for both Table 1 and Table 2.

Looking at the usr %, sys %, and tot %, it is clear that the rogue access point detector program of this report uses less CPU resources when running. Kismet has a multitude of other features as well as a terminal based GUI. Kismet also analyses every packet received, which is more than what the rogue access point detector program does. Some of this extra functionality does not assist detecting rogue access points, therefore wasting resources. The rogue access point detector program on the other hand solely focuses on detecting rogue access points and shows in the results as using half as much CPU.

Memory usage was compared between the two programs, represented by the rss/KB column. The value in this column is the physical memory used by the process and its child processes. The design of the rogue access point detector program shows that memory usage was not taken into account, for it uses an average of 137 MB of memory, compared to Kismet which uses only 3 MB. The rogue access point detector program has to start up an entire Java Virtual Machine (JVM) when running which takes up most of the memory footprint. Kismet is written in C/C++ and therefore has a much smaller memory footprint.

Comparing disk IO usage from the read KB/s and write KB/s columns, Kismet does not use any when running. The alerts created by Kismet are stored in memory and can optionally trigger an audible alert. The rogue access point detector software outputs its alerts to the standard out as well as a log file. The disk IO and size of the log file go up when more alerts occur.

Lastly, the number of milliseconds each process and its child processes ran per five second report duration is given. The usr-ms, sys-ms, and tot-ms columns show that Kismet runs for twice as long as the rogue access point detector program. This is due to the extra features that Kismet offers that don't necessarily aid in detecting rogue access points.

In the end, it is a tradeoff whether speed or extra functionality is required when picking a rogue access point detector. In the former case the rogue access point detector program introduced in this report has Kismet beat, but in the latter Kismet has the extra functionality needed to perform other tasks using the same program. The rogue access point detector program uses a magnitude more memory than Kismet, but uses half the CPU as Kismet. It is up to the Network Administrator to make the necessary tradeoffs in their situation.

5 Conclusion

5.1 Summary

It has been successfully shown that the rogue access point detector described in this paper is able to serve the purpose of detecting and alerting when a rogue access point starts advertising its presence. The report has also shown that the rogue access point detector program is comparable and in some ways better suited for rogue access point detection than Kismet. The rogue access point detector program allows a Network Administrator to automate the task of detecting and verifying that an organization's premises is free of rogue access points.

5.2 Relevance

Learning how to secure a wireless network is the next logical step after learning how the wireless technology works. There is only so much security that the 802.11 protocol offers to clients and network equipment. Taking security of a wireless network seriously involves scanning for rogue wireless access points [9]. The 802.11 protocol was not designed with rogue access points in mind. This is apparent because it is trivial for an attacker to set up an access point with the same ESSID of another access point and have wireless clients roam from the original access point to the rogue one.

5.3 Future Work

The rogue access point detector introduced in this report could further be improved in many ways. The fingerprinting of clients could become more advanced, involving reading more information from the 802.11 beacon. Values such as the beacon intervals, vendor information, encryption, ciphers and other tags can be used. Improving the fingerprinting of clients decreases the chances that rogue access points would be able to masquerade as a trusted access point.

Adding a learning mode to the rogue access point detector would enable administrators to configure the trusted access points much faster than manually entering it. The automatic configuration comes with a downside though: if the administrator were to neglect verifying that the obtained configuration contained only trusted access points then it is possible for a rogue access point to be marked as trusted.

Active detection is another method of rogue access point detection. The current rogue access point detector software could be extended to connect to all suspicious access points to further detect whether it is connected to the organization’s network or not. Administrators may want to know this information to determine whether the access point is located on premises or just outside. This can be achieved by connecting to the suspicious access point and trying to connect to a server that is only available on the organization’s network.

References

1. J. Leary and P. Roshan, “Wireless LAN Fundamentals: Mobility.” http://www.ciscopress.com/articles/article.asp?p=102282. Accessed: April 1, 2016.
2. J. Geier, “802.11 Beacons Revealed.” http://www.wi-fiplanet.com/tutorials/print.php/1492071. Accessed: April 1, 2016.
3. I. H. Saruhan, *Detecting and Preventing Rogue Devices on the Network*. Sans Institute, 2007.
4. R. Nayanajith, “802.11 Mgmt : Beacon Frame.” http://mrncciew.com/2014/10/08/802-11-mgmt-beacon-frame/. Accessed: April 8, 2016.
5. M. Idris, “Protect your Wireless Network from Evil Access Points!” https://github.com/moha99sa/EvilAP_Defender. Accessed: April 9, 2016.
6. H. Whiteman, “Security experts warn of dangers of rogue Wi-Fi hotspots - CNN.com.” http://www.cnn.com/2009/TECH/science/08/11/wifi.security.hackers/index.html?iref=24hours. Accessed: April 9, 2016.
7. L. Matthews, “Avast set up rogue access points at Mobile World Congress to prove people suck at security | News | Geek.com.” http://www.geek.com/news/avast-set-up-rogue-aps-at-mwc-to-prove-people-suck-at-security-1648170/. Accessed: April 9, 2016.
8. L. Higgs, “Free Wi-Fi? Beware of security risks.” http://www.usatoday.com/story/tech/2013/07/01/free-wi-fi-risks/2480167/. Accessed: April 9, 2016.
9. K. Zetter, “4 Years After TJX Hack, Payment Industry Sets Security Standards | WIRED.” http://www.wired.com/2009/07/pci/. Accessed: April 9, 2016.
10. L. Constantin, “This tool can alert you about evil twin access points in the area | PCWORLD.” http://www.pcworld.com/article/2905756/this-tool-can-alert-you-about-evil-twin-access-points-in-the-area.html. Accessed: April 9, 2016.