Identifying Rogue Access Point Attacks Using Probe Response Patterns and Signal Strength

Gabriel Ryan

January 2017

Identifying Rogue Access Point Attacks Using Probe Response Patterns and Signal Strength

***Abstract*— It has been over a decade since rogue access point attacks first emerged as part of the wireless security landscape. These attacks exploit weaknesses in the access point selection algorithms used by 802.11 clients. Despite improvements in wireless client design, rogue access point attacks have not only remained a relevant security threat, but have also grown increasingly sophisticated. In this paper, we will provide an overview of the weaknesses that make 802.11 susceptible to rogue access point attacks. Context will then be provided as we briefly describe the development of rogue access point attacks over the past decade. Existing research into rogue access point detection and mitigation will also be summarized. Finally, we will demonstrate the effectiveness of three techniques for detecting evil twin and karma attacks, two common types of rogue access point attacks. The first is a classic and widely used technique that uses whitelists to detect evil twin attacks. The second technique is a method for detecting evil twin attacks through statistical analyses of signal strength, and can be combined with the whitelist based approach to provide a more complete solution to rogue access point detection. The third and final technique is a method for detecting karma attacks by identifying access points that respond promiscuously to probe requests.**

I. Introduction

Despite of their age, rogue access point attacks remain as some of the most widely used and effective wireless attacks in use today. These attacks can usually be placed in one of two categories: evil twin attacks, which target a wireless network by exploiting the algorithms used by clients to roam between access points; and karma attacks, which target nearby wireless clients by exploiting the algorithms used for wireless network selection [1]. Both of these attacks are methods used for tricking or coercing wireless clients into associating with a wireless access point controlled by the attacker. This allows the attacker to execute man-in-the-middle attacks, steal login credentials for weak EAP configurations, and perform no-upstream attacks using a spoofed captive portal [2]. Although rogue access point detection has been a widely researched topic over the past decade, existing solutions either rely on assumptions about the implementation of the rogue access point that cannot be guaranteed, or do not account for edge cases that can be easily exploited by an attacker to evade detection.

II. Background

The 802.11 protocol defines two entities that can be present in a wireless network: base stations and access points. These entities can be used to build one of three kinds of wireless network, or basic service set (BSS). The first BSS configuration is known as an Independent Basic Service Set (IBSS), and consists of at least one base station. This type of configuration is commonly referred to as an "ad-hoc" network. Access points are not required within an IBSS. The second BSS configuration is the Mesh Basic Service Set, which consists of one or more mesh stations connected to zero or more mesh gates. The third BSS configuration is the Extended Service Set (ESS), which is also known as an "infrastructure" network. Infrastructure networks will be the primary focus of this document. Basic Service Sets within infrastructure networks consist of a collection of zero or more clients connected to an access point, and are identified by their Basic Service Set Identifier (BSSID). Each ESS is identified by its Extended Service Set ID (ESSID) [1].

The protocol defines two basic frame types that can be used by (base) stations and access points (APs) to locate and identify one another: probe frames and beacon frames. In an infrastructure network, beacon frames are periodically sent by access points to advertise their presence or capabilities to nearby stations. Beacons include information about the AP's supported data rates, encryption capabilities, and similar information. If ESSID broadcast is enabled, beacon frames also include the access point's ESSID. Networks can also be configured to not advertise their ESSID, in which case beacon frames are transmitted with the ESSID field set to an empty string. Probe frames come in two variations: probe requests and probe responses. Probe requests are sent by stations to determine what APs are within range, as well as what their capabilities are. Directed probe requests are used to query for specific networks by ESSID. Many stations periodically send directed probe requests for ESSIDs in their Preferred Network List (PNL), although this is behavior is not explicitly defined in the 802.11 standard [7]. A station's PNL is an ordered list that contains the ESSIDs of every network the station has connected to in the past. Broadcast probe requests serve a similar function to directed probe requests, and are used to query all access points within range. The ESSID of a broadcast probe request is set to an empty string. If an access point receives a probe request, and the ESSID value and supported rates denoted in the frame match those of the access point, then a probe response is issued to the station. Probe responses contain similar information to what can be found in beacon frames, including supported data rates, capabilities, and possibly the access point's ESSID [7][1].

The 802.11 protocol is very specific in how a station can join an ESS. Assuming no form of authentication is in place, stations associate with a network by sending an association request frame to an access point. The access point then responds with an association response, at which point the station becomes part of the access point's ESS. However, the protocol does not specify how the station should choose an ESS to connect to. Similarly, the protocol allows stations to roam by switching freely between access points that share the same SSID, but does not specify how these access points should be selected. Furthermore, although stations must be

trusted or authenticated by the ESS in order to associate with an access point, there is no requirement that an access point be trusted or authenticated by the station. This ambiguity was potentially intended to allow flexibility on the part of the 802.11 client. However, it has also lead to a class of attack that exploits the algorithms used by clients for ESS selection and access point selection during roaming [1].

The first such attack is the evil twin attack, which involves creating a clone of a legitimate access point. As previously mentioned, the 802.11 protocol allows stations to roam between access points in the same ESS. Stations rely exclusively on the 802.11 frame's SSID field to determine which ESS an access point belongs to. The protocol does not specify how stations should select an access point to connect to when multiple access points within the station's ESS are available [1]. Stations are typically optimized to respond to this scenario by selecting the access point within their chosen ESS that has the best signal strength and signal to noise ratio. Attackers can exploit this behavior to target an open wireless network by creating an access point with the same SSID as the target ESS. The attacker then boosts the signal strength of his or her access point such that it is stronger than all other access points on the target network. This causes all stations to roam to the attacker's access point, giving the attacker a man-in-the-middle between the stations and a network gateway [3].

Karma attacks are a second form of rogue access point attack that exploits the network selection process used by stations. In a whitepaper written in 2005, Dino Dai Zovi and Shane Macaulay describe how an attacker can configure an access point to listen for directed probe requests and respond to all of them with matching directed probe responses. This causes the affected stations to automatically send an association request to the attacker's access point. The access point then replies with an association response, causing the affected stations to connect to the attacker. This produces a man-in-the-middle similar to the one produced by an evil twin attack [1]. According to Ian de Villiers and Dominic White [2], modern stations are designed to protect themselves against karma attacks by ignoring directed probe responses from access points that have not already responded to at least one broadcast probe request. This led to a significant drop in the number of stations that were vulnerable to karma attacks until 2015, when White and de Villiers developed a means of circumventing such protections [2]. In White's and de Villiers' improved karma attack, directed probe responses are used to reconstruct the PNLs of nearby stations. When a broadcast probe request is received from a station, the attacker's access point responds with an arbitrary SSID from the station's PNL. This process occurs simultaneously to the original karma attack, and was found to greatly improve its effectiveness [2].

To summarize, evil twin and karma attacks exploit the mechanisms used by stations to roam from one access point to another within a single ESS, as well as the mechanisms used to determine which available ESS to connect to. Although karma attacks have had to evolve in recent years in order to account for changes in 802.11 client behavior, all forms of rogue access point attack remain highly effective despite relying on principles and techniques that were introduced over a decade ago. Solutions for detecting, locating, and preventing these types of attacks are in high demand. To this end, substantial research has been focused on this area. In the following sections we will explore existing research into rogue access point attack detection. We will then replicate a classic whitelist based approach for detecting evil twin attacks in section 4. In section 5 we will then expand upon this approach with a technique for detecting evil twin attacks through a statistical analysis of signal strength. Finally, section 6 will describe a reliable technique for detecting karma attacks using overlapping probe responses.

III. Related Work

Evil twin and karma attack detection have been the topic of considerable research. Techniques to detect evil twin attacks using BSSID whitelisting have existed since at least 2006, and are outlined in a SANS publication by Larry Pesce [3]. In this approach, a wireless network is protected by taking an inventory of every access point that is permitted to operate using the network's SSID. A whitelist is then created containing the BSSIDs of these access points. An 802.11 client is then placed into radio frequency monitor mode (RFMON) and allowed to sniff probe response frames from nearby access points. If a probe response frame is found to have the SSID of the network being protected, but a BSSID that is not in the whitelist, it is presumed to have been transmitted by an evil twin [3]. This approach will be expanded on in further detail in section 4, since it works well when used in conjunction with the signal strength based approach described in section 5. It should be noted that this is not a complete solution to evil twin detection, as an attacker can simply create an evil twin with the BSSID of a whitelisted access point. In this case, the evil twin would remain undetected.

A radically different approach was taken by researchers Fabian Lanze, Andriy Panchenko, Ignacio Ponce-Alcaide, and Thomas Engel of the University of Luxembourg. They observed that rogue access point attacks are most commonly executed using high level software and external wireless interfaces originally intended to serve as base stations. They reasoned that because of this, detecting evil twin attacks is largely a matter of distinguishing between these "soft" access points and traditional access points managed by firmware. They developed three effective techniques for doing so [4].

However, their approach does not account for rogue access point attack tools that are managed by firmware.

Jana and Kasara used a fingerprint based approach in which access points were given a signature derived from clock skew. The baseline fingerprint for each legitimate access point was first derived using a Markov Chain based algorithm for analyzing packets sniffed at the network gateway. Their technique was one of the first to account for cases in which the attacker spoofs the BSSID (MAC address) of the rogue access point [6].

Han et al. developed a technique for detecting rogue access points by analyzing round trip time between stations and their DNS servers. This approach was unique at the time in that instead of attempting to detect rogue access points from the perspective of a legitimate access point, it focused on detecting rogue access points from the perspective of a base station. Han et al. demonstrated that if a difference in the number of hops between the base station and DNS server[1] is detected, it can be assumed that the base station is connected to a rogue access point. This technique relies on the assumption that the rogue access point is acting as a bridge between the target base stations and the target network. It does not account for captive portal attacks in which no upstream interface is used, does not account for situations in which a WDS is used instead of a wired DS, and does not account for attacks in which the attacker bridges traffic between the rogue access point and a secondary network gateway such as a mobile hotspot [7].

Although all of these techniques provide some degree of rogue access point detection, none of them offer a complete solution. Regardless, the whitelisting approach to evil twin detection is worth exploring in greater detail. It is easy to implement, and is highly effective in cases where the attacker does not spoof the BSSID of a whitelisted access point. To account for these cases, it can be combined with a more complete method of evil twin detection such as the one described in section 5. Despite the fact that the signal strength based algorithm covered in section 5 can operate independently, using it to augment the otherwise reliable and efficient approach of BSSID whitelisting provides a more well-rounded and complete algorithm for evil twin detection. Pairing these two algorithms with the karma attack detection technique described in section 6 provides a solution for efficiently detecting both evil twin and karma attacks without the need for administrative access to the network that is to be protected.

IV. Detecting Evil Twin Attacks - Whitelisting

Recall that in an evil twin attack, the attacker creates a wireless access point using the SSID of the target ESS. In most cases, the attacker's BSSID is irrelevant to the success of the attack, as it is not typically used by the affected stations to determine whether the access point is a valid member of the network. Exceptions include cases in which stations maintain a whitelist of BSSIDs that are known to belong to a particular ESS. Given that the BSSID is taken from a 48-bit address space, it is virtually impossible that the BSSID of the evil twin will match the BSSID of the rogue access point, unless the attacker intentionally configures it to be this way. This makes it possible to use BSSID whitelisting as a means of detecting evil twin attacks. Whitelist based algorithms are a classic approach to evil twin detection outlined in multiple sources, most notable of which is a 2006 SANS publication by Larry Pesce [3]. To protect a network against evil twin attacks using this approach, a whitelist is created containing the SSID and BSSID of every ESS that is part of that network. Packets are then captured using a wireless network interface placed in monitor mode. Any packet that is not a beacon frame or directed probe response is discarded. The SSID and BSSID fields of each captured probe response are checked against the whitelist. If a packet is found to have an SSID that is in the whitelist, but a BSSID that is not, it is assumed to have been transmitted by an evil twin [3].

Algorithm 1 whitelist based algorithm for detecting evil twins

```
1:  whitelist = new HashMap()
2:  for each access point p in network A do
3:      ssid = p.ssid
4:      bssid = p.bssid
5:      if not whitelist.iskey(ssid) then
6:          whitelist[ssid] = new Set()
7:      end if
8:      whitelist[ssid].add(bssid)
9:  end for
10: r = probe_responses.next()
11: while r do
12:     ssid = r.ssid
13:     bssid = r.bssid
14:     if whitelist.iskey(ssid) and bssid not in whitelist[ssid] then
15:         send_alert()
16:     end if
17:     r = probe_responses.next()
18: end while
```

a.

Fig. 1. – a whitelist based algorithm for detecting evil twin attacks

In the algorithm shown in figure 1, a wireless network is represented using a (hash) map. The set of all SSIDs used in the wireless network is used as the map's set of keys. Each value in the map corresponds to the set of all access points permitted to use the key as their SSID. Probe responses from

[1] The DNS server is assumed to be internal to the organisation.

nearby wireless access points are then collected using a packet sniffer. For each probe response received, the packet's SSID is checked against the list of keys in the map. If the SSID is a key in the map, the packet's BSSID is checked against the set of all BSSIDs whitelisted for the SSID. If the BSSID is not in the set, then the algorithm yields an alert.

In order to test its effectiveness, this algorithm was implemented as a proof of concept Python script using the Scapy networking library. The script was loaded onto a Raspberry Pi Model B, which was equipped with a TP-Link wn722n external wireless adapter using Atheros drivers. A wireless access point was placed two feet away from the Raspberry Pi, and the SSID and BSSID of the access point were added to the Pi's whitelist. A laptop equipped with hostapd, a software program used for creating wireless hotspots, was placed 50 feet from the access point. The laptop was also equipped with a TP-Link wn722n adapter. The Python script was used to sniff packets for a period of five minutes, throughout which no evil twin attacks were detected. At the end of this five-minute period, the script continued to run while an evil twin attack was launched against the access point using the laptop equipped with hostapd and a TP-Link wn722n external wireless adapter. The BSSID of the laptop based evil twin was deliberately set to a different value than that of the access point. Upon capturing probe responses from the evil twin, the algorithm caused the Raspberry Pi to issue an alert that indicated the presence of an attack. These results demonstrate that it is possible to detect evil twin attacks using whitelist based algorithms. However, these results do not account for scenarios in which the attacker sets the BSSID of the evil twin to match the BSSID of the target access point. A means of accounting for this case is provided in the next section.

V. Detecting Evil Twin Attacks – Signal Strength

The approach outlined in section 4 is effective and reliable for cases in which the attacker does not spoof the BSSID of the evil twin. However, whitelisting will fail to detect evil twins that are created using the BSSID of a legitimate access point on the targeted network. This means that the whitelist based algorithm must be augmented to account for this edge case if it is to provide a more complete solution to evil twin detection. One means of doing this is by monitoring the TX values of sniffed probe response frames. Suppose a sensor is placed a fixed distance from a wireless access point. Assuming that both the access point and the sensor remain stationary, the access point's signal strength as read from the sensor should vary at a consistent rate over a period of time. That is to say, while the TX values of packets received by the packet sniffer will vary, this variance should fall within a recognizable threshold.

Now let us consider what would occur if an attacker were to create an evil twin of the legitimate access point. For the following scenarios, let us assume that the transmit power of the evil twin and the legitimate access point are exactly the same. If the packet sniffer is placed significantly closer to the access point than to the evil twin, then any packets received from the evil twin would have a noticeably lower TX value than those received from the legitimate access point. Conversely, if the packet sniffer is placed closer to the evil twin than to the access point, the opposite would be true. Therefore, it follows that any packet received by the sniffer from the evil twin would have a TX value that falls outside of the recognizable threshold set by the legitimate access point. The only situation in which this would not be true would be when the evil twin and the legitimate access point are equidistant from the packet sniffer, creating a blind spot. This edge case can be accounted for by using two packet sniffers: one located immediately adjacent to the access point, and another located a significant distance away. Even if the transmit power of the evil twin and legitimate access point are different, the only thing that changes is the location of the blind spot.

Algorithm 2 whitelist algorithm augmented with signal strength analysis

```
1:  r = probe_responses.next()
2:  while r do
3:      ssid = r.ssid
4:      bssid = r.bssid
5:      tx = r.tx
6:      if whitelist.iskey(ssid) and bssid not in whitelist[ssid] then
7:          send_alert()
8:      else
9:          upper_bound = calibration_table[ssid][bssid].upper
10:         lower_bound = calibration_table[ssid][bssid].lower
11:         if tx > upper_bound or tx < lower_bound then
12:             send_alert()
13:         end if
14:     end if
15:     r = probe_responses.next()
16: end while
```

b.

Fig. 2. – An algorithm that uses signal strength to detect evil twin attacks

This conclusion allows us to augment our previous algorithm by setting an upper bound and lower bound for the TX values of packets collected from network A. For each packet received, the value of the packet's SSID field is checked against the list of keys in the map. If the SSID is a valid key, the value of the packet's BSSID field is compared against the set of BSSIDs mapped to by the SSID. As in the original algorithm, the packet is assumed to have originated from an evil twin if its BSSID is not in the set. However, unlike the previous algorithm, an additional check is performed if the BSSID is a member of the set. If the TX value of the packet is not between the upper and lower bounds set for that BSSID, the packet is assumed to have been transmitted by an evil twin. The augmented evil twin detection algorithm is shown in *figure 2*.

Algorithm 4 calibration algorithm

```
for each access point t in network A do
    ssid = t.ssid
    bssid = t.bssid
    if calibration_table.haskey(ssid) == false then
        calibration_table[ssid] = new HashMap()
    end if
    calibration_table[ssid][bssid] = t
    packets = t.capture_n_packets()
    num_packets = packets.length
    tx_list = [p.txforpinpackets]
    mean = sum(tx_list)/tx_list.length
    max_dev = max(|el − mean| for el in tx_list)
    upper_bound = mean + max_dev * 2
    lower_bound = mean − max_dev * 2
    calibration_table[ssid][bssid].upper = upper_bound
    calibration_table[ssid][bssid].lower = lower_bound
end for
```

c.

Fig. 3. – An algorithm that uses signal strength to detect evil twin attacks

For the algorithm shown in *figure 2* to succeed in identifying evil twins, a reliable method of establishing the upper and lower bounds for allowable TX values is needed. One method is to use a calibration algorithm such as the one shown in *figure 3*. The calibration algorithm works by capturing probe responses sent by a specified access point over a period of time.

To test the effectiveness of this second algorithm, a sensor was placed two feet away from a legitimate wireless access point controlled by the researcher. The sensor was built using a Raspberry Pi and a TP-Link wn722n wireless adapter. The second evil twin detection algorithm, as well as the calibration algorithm, were both implemented as Python scripts and loaded on the sensor. The laptop used in section 4 was placed 50 feet away from the wireless access point at a parallel angle to the sensor. The calibration script was run on the sensor to establish lower and upper bounds to create a threshold of allowable TX values. The evil twin detection script was then run on the sensor for a five-minute period, throughout which no evil twins were detected. After five minutes, an evil twin was created using hostapd on the laptop. The BSSID of the evil twin was set to match the BSSID of the legitimate access point. This caused the sensor to issue an alert that an evil twin was nearby.

Based on the results of this experiment, we can conclude that it is possible to detect evil twin attacks through the statistical analysis of signal strength. More research is necessary to study the effectiveness of this approach under varying environmental conditions. It is likely that a more complex statistical model would be required to account for circumstances such as RF reflectors and power fluctuations, as these factors could cause significant variation in TX values from a legitimate access point. Additionally, monitoring 5GHz wireless networks for the presence of evil twin attacks can be more complicated due to regulatory standards that mandate the use of Dynamic Frequency Selection (DFS) on some or all 5GHz channels. DFS is a protocol introduced in 802.11h that is designed to prevent 802.11 networks from interfering with radar systems operating within the 5.250-5.75 GHz range. An access point using DFS must be able to detect the presence of radar on its operating channel, and switch channels should this occur. Since the legal limits for transmit power tend to vary from channel to channel, and access points are often configured to transmit near maximum capacity, the channel switching behavior introduced by DFS cause noticeable fluctuations in TX power. Therefore, the methods outlined in in this section could be prone to false positives when used on DFS channels. As such, administrators who wish to use these techniques should avoid DFS channels when running 5Ghz networks in areas where frequent radar interference is likely to occur.

VI. Detecting Karma Attacks

In a karma attack, the rogue access point replies to all probe requests it receives by sending a valid probe response. This causes nearby wireless clients that actively probe for the networks on their PNL to connect to the rogue access point. Karma attacks do not impersonate a single access point, which makes detection algorithms that rely on whitelisting largely ineffective. However, karma attacks involve blindly responding to all probe requests received. This holds true regardless of whether or not the probe requests are sent from a legitimate client. Therefore, it is possible to create a karma honeypot by broadcasting probe requests for multiple unique SSIDs. Nearby rogue access points performing karma attacks will respond to more than one of these requests using the same BSSID. By doing so, they will reveal their existence. An algorithm that relies on this assumption is shown below, in *figure 4*.

Algorithm 3 algorithm to detect karma attacks

```
lookup_table = new HashMap()
r = probe_responses.next()
while r do
    ssid = r.ssid
    bssid = r.bssid
    if lookup_table.iskey(bssid) and ssid not in lookup_table[bssid] then
        lookup_table[bssid].add(ssid)
        send_alert()
    else
        lookup_table[bssid] = new Set()
        lookup_table[bssid].add(ssid)
    end if
    r = probe_responses.next()
end while
```

d.

Fig. 4. – An algorithm that uses signal strength to detect evil twin attacks

The effectiveness of this algorithm can be improved by periodically broadcasting probe request packets with the SSID field set to a random string. If these probe requests are broadcasted in parallel with the algorithm shown in *figure 4*, it allows the algorithm to detect karma attacks even when no devices nearby are actively probing for networks on their PNL.

It can be argued that this approach will not work in cases where the rogue access point switches its BSSID between each probe response. However, approaches in which the BSSID of the access point is cycled between each probe request suffer from a fundamental constraint: the rogue access point must be able to respond to probe requests for an arbitrary number of SSIDs. For the attack to be reliable, it must be assumed that each probe response sent by the AP will result in a subsequent association attempt by the station to which the response is directed. To use a new BSSID for each probe response sent, the access point must be restarted between requests or multiple VAPs must be used. Restarting the access point between requests means that affected stations can only remain associated with the AP briefly between each probe request received. Problems also arise if VAPs are used. In the worst case scenario, the access point receives *N* probe requests in quick succession. This causes the AP to issue *N* probe responses and create *N* VAPs. The management traffic from these VAPs alone would be enough to significantly decrease the bandwidth of on the AP's operating channel, affecting both the AP and its neighbors. Additionally, VAP creation is a resource intensive task, making the creation of numerous VAPs in quick succession difficult on most systems. These are just some of the obstacles that make BSSID switching within a Karma attack theoretically interesting, but infeasible in any practical sense as of this time.

This algorithm was tested by placing a sensor 2 feet away from a wireless access point controlled by the researcher. A laptop equipped with hostapd-mana, a modern karma attack software tool, was placed 50 feet from the access point in the same direction as the sensor. The sensor was built using a Raspberry Pi Model B, and given a TP-Link wn722n external wireless adapter to inject packets. The laptop was also given a TP-Link wn722n with which to create an access point. A Python script implementing the algorithm described in the previous section was loaded onto the sensor. The Python script implementing algorithm 3 was run on the sensor. A karma attack was then initiated from the laptop using hostapd-mana. The rogue access point created using the laptop and hostapd-mana responded to each probe request sent by the sensor, causing the sensor to issue an alert. Based on the results of this analysis, it can be concluded that Karma attacks can be detected by identifying wireless access points that respond to multiple distinct SSIDs. It can also be concluded that rogue access points used for karma attacks will respond to programmatically sent probe requests, allowing them to be detected without the presence of nearby client probing.

VII. Closing Thoughts

Rogue access point attacks continue to exist as a primary threat to enterprise wireless networks. Nearly all forms of WPA2-EAP networks can be compromised using evil twin attacks, with networks protected by EAP-TLS being the notable exception. To compromise an EAP-PEAP network, for example, an attacker merely creates an evil twin of one of the network's access points. When the associated clients connect with the attacker's AP, the attacker's radius server presents them with a bogus certificate. Any client that does not reject the certificate will then attempt to authenticate with the attacker's radius server using credentials for the target network. The attacker can then use the perform an offline dictionary attack using the captured challenge and response to obtain valid network credentials. With the exception of EAP-TLS, other forms of EAP can be compromised using similar methods.

Although an evil twin attack may not be a viable option for an adversary seeking to compromise a network protected by EAP-TLS, rogue access point attacks can still be used to attack the network's clients directly. An attacker could initiate a Karma attack or evil twin of a network on the client's PNL, then deauthenticate clients from the EAP-TLS network. The attacker could then perform a man-in-the-middle attack against the affected clients, stealing sensitive data such as accounts credentials that could be used to pivot into the network through other means.

Fortunately, the techniques used to implement evil twin and karma attacks represent atypical behavior that can be algorithmically identified. Evil twin attacks can be detected using whitelist based algorithms, provided that the evil twin does not share a BSSID with a legitimate access point. Evil twin can also be identified through the statistical analysis of signal strength. More research is needed to improve the effectiveness of this approach under adverse environmental conditions. Identifying evil twin attacks through the analysis of packet round-trip time is another option that warrants exploration. Additionally, Karma attacks can be detected by monitoring for one-to-many relationships between a single BSSID and multiple SSIDs. The effectiveness of this approach can be improved by programmatically sending probe responses for multiple unique SSIDs and analyzing the responses.

VIII. Acknowledgements

Special thanks should be given to Mr. D Bastone for his guidance and support throughout the project, as well as Mr. M Hopkins for his thoughtful and constructive recommendations.

Additional thanks should be given to Mr. S Drew and Mr. D White for their inspiration, as well as the BSides Las Vegas organization for providing the author with a live testing environment.

About GDS Labs

Security Research & Development is a core focus and competitive advantage for GDS. The GDS Labs team has the following primary directives:

- Assessing cutting-edge technology stacks
- Improving delivery efficiency through custom tool development
- Finding & responsibly disclosing vulnerabilities in high value targets
- Assessing the impact to our clients of high risk, publicly disclosed vulnerabilities

The GDS Labs R&D team performs security research, with areas of current focus including mobile application security, embedded systems, radio communications, and cryptography. GDS also participates in many security related organizations and groups such as OWASP and the NodeJS Security Project. GDS Labs is a value add service that our clients benefit from on virtually every engagement that we perform.

About Gotham Digital Science

Gotham Digital Science (GDS) is a specialist security consulting company focused on helping our clients find, fix, and prevent security bugs in mission critical network infrastructure, web-based software applications, mobile apps and embedded systems. GDS is also committed to contributing to the security and developer communities through sharing knowledge and resources such as blog posts, security tool releases, vulnerability disclosures, sponsoring and presenting at various industry conferences. For more information on GDS, please contact info@gdssecurity.com or visit http://www.gdssecurity.com.

References

[1] Dai Zovi D. Macaulay S. (2005) Attacking automatic wireless network selection. Proceedings from the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2005.

[2] White D. de Villiers I. (2015) Improvements In Rogue AP Attacks – Mana (1/2) [Web log post]. Sensepost.

[3] Pesce L. (2006) Discovering Rogue Wireless Access Points Using Kismet and Disposable Hardware. SANS Institute.

[4] Lanze F, Panchenko A, Ponce-Alcaide I, Engel T. (2015)Hacker's Toolbox: Detecting Software-Based 802.11 Evil Twin Access Points. University of Luxembourg Interdisciplinary Centre for Security, Reliability and Trust.

[5] Jana S. and Kasera S. K. (2008). On fast and accurate detection of unauthorized wireless access points using clock skews. MobiCom'08, September 14–19, San Francisco, California, USA

[6] Han, H., Sheng, B., Tan C. C., Li, Q. and Lu, S. (2009). A Measurement Based Rogue AP Detection Scheme. In Proc. IEEE INFOCOM'09

[7] "IEEE Standard for Information Technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std. 802.11-2012 (Revision of IEEE Std 802.11-2007),* p 11,