WiFiHop - Mitigating the Evil Twin Attack through Multi-hop Detection

Diogo Mónica and Carlos Ribeiro

Instituto Superior Técnico / INESC-ID Lisboa,
Rua Alves Redol 9, sala 605, 1000-029, LISBOA
{diogo.monica,carlos.ribeiro}@ist.utl.pt
http://www.gsd.inesc-id.pt

Abstract. Public hotspots have undeniable benefits for both users and providers. Users get ubiquitous internet access and providers attract new potential clients. However, the security mechanisms currently available (e.g. WEP, WPA) fail to prevent a myriad of attacks. A particularly damaging attack to public WiFi networks is the evil twin attack, where an attacker masquerades as a legitimate provider to mount wireless interposition attacks. This paper proposes WiFiHop, a client-sided tool that leverages the intrinsic multi-hop characteristics of the evil twin attack, to detect it. The proposed tool is technology independent (e.g. network bandwidth or latency), and detects the attacks in real time (i.e. before any user traffic is transmitted). It works with both open and encrypted networks. This tool was tested in a real-life scenario, and its effectiveness demonstrated.

1 Introduction

Wi-Fi networks have become a ubiquitous technology. These networks enable users to access the internet at home, at work, and even when traveling, but they are vulnerable to a number of threats, mainly because wireless access point (AP) operators often don't take the time to activate the adequate security features. The theft of private information is, therefore, becoming a growing concern. Users accessing the internet with wireless devices in public places (e.g. cafes or airport terminals) are particularly susceptible to these attacks.

An increasingly common strategy for the theft of private information is the evil twin attack, which consists of having an unsuspecting user automatically associating to an AP under the control of the attacker (a rogue AP). This access point is configured to mimic a legitimate access point (for example, by copying the legitimate network's SSID name), and enables attackers to eavesdrop all wireless communications done by the victims. Due to these two characteristics, these rogue access points are usually called *evil twin* APs.

There are essentially three different strategies for attackers to lure victims into connecting to their rogue AP. The first one is by having a higher signal strength than the other AP. This strategy works, since several operating systems choose the AP with the strongest signal strength, even when several APs with the same SSID are available. Also, users tend to choose the network with the higher signal strength when manually choosing a network to connect to. The second strategy uses the automatic re-association

V. Atluri and C. Diaz (Eds.): ESORICS 2011, LNCS 6879, pp. 21–39, 2011.
© Springer-Verlag Berlin Heidelberg 2011

feature that several end-user systems provide. These systems have preferred network lists, containing the SSID names of the networks a user has previously connected to in the past. To exploit these lists, the attacker simply choses the evil twin AP SSID name to be one of the most commonly used SSID names (e.g. linksys), and waits for victims to connect. Finally, the third strategy involves using a denial-of-service attack against 802.11 networks. Attackers use well-known vulnerabilities in 802.11 to prevent a client from initially associating to a legitimate AP, or even to disassociate clients already associated [13]. The loss of connectivity resulting from the continuous disassociations, forces users to select other available wireless networks. This strategy can be highly effective, especially when combined with one of the previous two.

The evil twin attack is usually launched at public places where open-access WiFi networks are available. Locations like airports or cafes, are ideal, since there is no way for the users to distinguish rogue from legitimate APs [9]. Using an evil twin AP, attackers can effectively intercept all kinds of sensitive data such as passwords, credit-card information or even launch man-in-the-middle and phishing attacks. The malicious potential of this attack, together with the ease in configuring and deploying rogue APs, makes this attack a serious threat to wireless networks. This attack is particularly hard to trace, since it may occur only for a short amount of time. Depending on the objective of the attacker, after a few minutes of operation, the attack can be terminated, causing nothing more than a network disconnection for the victims (something somewhat common in wireless networks). In this short time frame, the attacker may already have compromised user's sensitive information.

The rest of this paper is organised as follows. Section 2 introduces the existing solutions to the generic problem. In Section 3 we detail the exact setup and assumptions addressed in the paper. Section 4 describes in detail the operation of our evil twin attack detection mechanism. In Section 5, our algorithm implementation is described. Finally, the results of the implementation of our mechanism are presented in Section 6. Section 7 concludes the paper.

2 Related Work

Existing solutions are mainly focused on the detection of rogue APs by the network administration, and not by the users themselves. One of the original ways of detecting these rogue APs relied on the manual verification of the available APs by a network administrator, using network enumeration tools such as Netstumbler [2]. This proved to be ineffective, since manual scans are time-consuming and expensive, and were therefore conducted infrequently. Since then, several automated systems have been proposed [1,8,6,11,10]. These solutions monitor the wireless medium and other types of information gathered at the network routers/switches, and compare them with a known authorisation list. For example, AirDefense [1] uses a combination of radio-frequency sensors jointly with an intrusion detection server to capture, process, and ultimately correlate network events, in search for APs with unknown "fingerprints". In other works [10,11], special diagnostic software was installed on mobile clients to perform wireless medium monitoring, helping the detection of rogue APs. Several variations, such as using sensors instead of sniffers to scan the wireless medium, have also been proposed ([6]). However, most of these solutions suffer from some, or all, of the following

problems: they do guarantee a complete coverage of the network (required to ensure effective rogue AP detection); they may flag a normal AP (e.g. the access point of a nearby coffee shop) as a rogue AP; they do not work for rogue APs that possess authentication mechanisms such as WEP and WPA; and finally they may access unauthorised networks in the process of testing all the available APs in the vicinity.

A different line of work has also been pursued, where researchers attempt to distinguish wireless, from wired hosts, by analysing wired network traffic ([25,18,26,24,28,20,12,23]). The RIPPS system [18], for example, deduces wireless connectivity from the existing wired network traffic. The objective of RIPPS is to detect rogue APs without using wireless sensors. It uses active network traffic conditioning together with passive packet timing analysis. Wei et al. [26] use wireless traffic characteristics to distinguish wireless nodes. They present two detection algorithms that apply sequential hypothesis tests to packet-header data. These algorithms are able to detect wireless TCP traffic by considering specific properties of the 802.11 CSMA/CA mechanism. A similar work was done by Xie et al. [27], where the TCP jitter was proposed as the distinguishing characteristic between wired and wireless nodes. Several works [20,14] propose the analysis of the differences in inter-packet spacing to achieve the same objective. Other characteristics of the wireless traffic flows, like the client-side bottleneck bandwidth or the round trip time of network traffic, are also used in [16] and [17] (respectively). However, this general scheme assumes that there is a way to analyse all the network traffic, which might not be practical, and severely limits the scalability of these systems.

Finally, there are also some hybrid solutions like Yin et al. [28], where both sniffers and wired traffic analysis are used. In this particular case, they use an intrusion detection mechanism that uses a verifier on the internal wired network to send test traffic towards wireless edges. This mechanism can detect rogue access points, by detecting the relay of the test packets to the wireless edge. Liran Ma et al. [17] also proposed a hybrid solution, by correlating anomalies using both wired and wireless scans, being able to detect not only unauthorised but also compromised APs.

In these administrator-oriented solutions, a user must trust the network. Also, most of these solutions are not real-time, allowing short time attacks to remain undetected. Secondly, even if the detection is done in a timely fashion, many users can still fall prey to the attack, since there is no automated way of denying access to the evil twin APs or even warn the users of the attack. Our work shifts the usual paradigm, and empowers users with a tool that allows them to detect if an evil twin attack is being launched, before they actually start using the network.

To the best of our knowledge, only one author employs a client-side method, designed for evil twin attack detection. Song et al. [21], propose ETSniffer, using timing measurements to distinguish a one-hop from a multi-hop setting. However, as stated in [18], these timing measurements are technology dependent. With the increase in wireless networks transmission rates (e.g. 802.11n), differences in delay between wireless and wired settings will fade, or at least, vary. This means that a wireless node may become indistinguishable from a wired node, or, in the particular case of ETSniffer, a multi-hop setting, indistinguishable from a one-hop setting in a faster technology.

Our solution differs from previous work in that it does not depend on timings to detect a multi-hop setting. Instead, detection is based on the behaviour of the legitimate AP, thwarting the attacker from evading the mechanism. Additionally, we do not require the knowledge of an authorisation list, and allow the user to proactively test the network, prior to using it. The fact that detection is done by the users makes this an efficient and cost-effective solution, where no modifications to the client hosts are required.

3 Problem Statement

We assume the existence of an 802.11 wireless LAN device, capable of operation in monitor mode, being operated by a user that wishes to access the internet through a free hotspot. This user has no knowledge of the infrastructure, and cannot, thus, verify the authenticity of any available access points in the vicinity. We further assume that the user is in radio range of the legitimate access point, where the wired connection to the internet is located. In fact, this scenario describes almost all public locations where a free hotspot is available, such as cafes or airports.

In normal operation, a user in the vicinity of a legitimate AP will associate directly to it, and access the internet. However, in the case where there is a malicious attacker launching an evil twin attack, the user might be fooled into associating to the evil twin AP, instead of directly connecting with the legitimate one (Figure 1). This evil twin AP will also allow internet access to the user, by forwarding all the information received from the user, wirelessly, to the legitimate AP. It will, therefore, be capable of intercepting all the user information, without being detected. Notice that, in both scenarios, the legitimate AP is within range of the user.

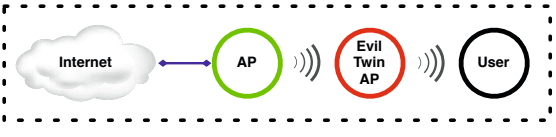

Fig. 1. Illustration of the problem being addressed

Our objective is to provide a convenient and usable technique to detect the existence of an evil twin attack. The detection scheme will be required to possess the following properties:

- Operation not detectable by the attacker;
- Capable of operation in encrypted networks;
- Independent control of both the probability of detection p_D and of false alarm p_{FA};
- Non-disruptive operation;
- User-sided operation.

The rationale for requiring most of these properties is clear. A note must, however, be made, concerning the last property listed (*user-sided operation*). The detection algorithm to be developed is intended to constitute a tool which users may ubiquitously

possess, to be used automatically whenever the user joins a public hotspot, or in any instance where it may be considered necessary. The use of solutions such as end-to-end VPNs, is much more complex in terms of implementation/setup and still leaves users vulnerable to layer 2 and denial-of-service attacks.

4 Detecting the Evil Twin AP

The presence of an evil twin AP will be detected using the fact that, when such an attack is in place, the user's data must transit the wireless channel between the evil twin and the legitimate AP (see Figure 1). If we can detect the existence of this extra wireless hop, we will then have attested the presence of the evil twin AP.

The basic overall scheme for detecting the existence of a multi-hop setting between a user and the internet is the following (details will be addressed later on, and will depend on the particular type of network under analysis): the user sends a watermarked packet (or packet sequence) to the internet, through the access point to which it is currently associated with; the watermark signature is known only to the user; after sending this packet in the channel associated with its AP connection, the user listens to a different wireless channel, and tries to detect the presence of the watermark in the traffic passing through that wireless channel. If an evil twin attack is being launched, the watermark will necessarily appear on the wireless link between the evil twin and the legitimate APs, and the attack will, therefore, be detected, if the user repeats the procedure for every one of the available wireless channels. We called this generic scheme WiFiHop.

Since the time taken for the wireless access point to retransmit the watermarked packet (the store-and-forward delay of the AP) is orders of magnitude lower than the time taken for a user to switch channels and begin parsing wireless traffic, detection of the watermark cannot be done on the outward path, since by the time the user is ready to detect the watermark, the packet has already been forwarded to the legitimate AP. We therefore send the watermark packet to an external server on the internet, which we assume not to be malicious, having the packet being replayed back to the user a pre-defined number of times. The watermark will then be detected on the incoming, returning path, when being forwarded to the evil AP by the legitimate one. This scheme not only allows the user to properly switch channels and initiate monitoring mode (if that is the case), but it also allows the option of simply requesting the watermark from

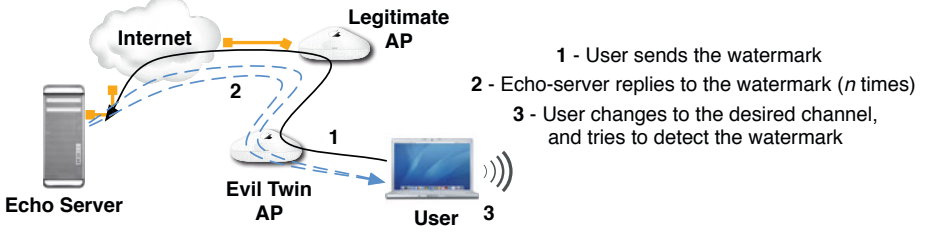

Fig. 2. The WiFiHop mechanism

the server, avoiding the need for its outward transmission, a scheme which will be used in the later parts of the paper. Figure 2 depicts this evil twin detection mechanism.

There is however, one important issue that has to be taken into account. We have to consider the possibility of existence of an encrypted channel between the evil twin AP and the legitimate AP. While most hotspots do not have encryption in place, wireless security mechanisms such as WEP and WPA, are commonplace nowadays. In the case where the link between the evil twin and the legitimate AP is protected with one of these security mechanisms, the user will be unable to detect a watermark embedded into the payload of the encrypted packet. We will, thus, provide two different solutions for detecting the watermark in these two scenarios (plain and encrypted channels): Open WiFiHop and Covert WiFiHop.

In both solutions, we will need to take into account the possibility of packet loss. In multi-hop wireless networks, specially with high traffic load, packet losses are frequent. This possibility of packet loss will, therefore, be accounted for in the proposed scheme.

Finally we note that there are legitimate uses for multi-hop schemes (such as range expansion). While false positives may occur when both APs (the original and the expander) are in range, the user will simply be instructed to use the AP that is directly connected to the internet. If only one of the APs is in range no false positive will occur.

4.1 Open WiFiHop

In Open WiFiHop, we assume that there is no encryption between the evil twin AP and the legitimate AP. Watermark detection can, thus, be easily done. The user creates a random bitstring, and sends it to the echo-server. Then, it tries to find the exact same pattern in the payload of all the packets being sent in one of the alternative wireless channels. The test is repeated for every wireless channel, other than the one being used in the association between the user and the service providing AP.

In this particular case, and since we can increase the size of the random bitstring, there is almost no possibility of having false positives (false alarm). A false alarm would happen if the random bit string chosen by the user happened to be present, by chance, in any wireless packet other than the echo-server's answer. However, by choosing a bitstring of any reasonable size (e.g. 128 bits), we can bring this probability arbitrarily close to 0. This aspect will be discussed again, when discussing the detection performance of the algorithm.

However, there still is the possibility that either the client's request or the reply watermarked packets are lost in transit, something that would make Open WiFiHop return a false negative (miss). There are four situations that will translate in Open WiFiHop not detecting the watermark: packets being lost between the echo-server and the legitimate AP; packets being lost in the air, between the legitimate AP and the evil twin AP; packets being delayed more than the time window allocated to the test; and finally, reply packets not being detected by the sniffer (the user network card). Of these, the last one is conspicuous, since it creates a considerable asymmetry between the probabilities of losing a request, and the probability of not detecting a reply. As will be seen, the efficiency of the sniffer can vary widely (we used both TCPDump [5] and Scapy [4]), but will, in all cases, constitute a major source of loss of reply packets. We will, therefore, consider two different probabilities of packet loss, one for outgoing packets (requests),

p_{plo}, and one for incoming packets (replies), p_{pli}. To be able to control the statistical performance of the method in this packet loss environment, clients will repeat the request c times for each wireless channel, and the server repeats the watermarked packet n times, in response to each received request.

Statistical Performance. It is assumed that the observation window (t) is chosen to be wide enough so as to render negligible the probability that the transmitted packets fall outside the window (which might otherwise occur, due to excessive latency in the network). The choice of t will be addressed later on in the paper. Since the number of bits constituting the watermark can be arbitrarily long (within the restriction imposed by the network's MTU), the probability of a particular pattern appearing in an external packet (p_{ra}) is, for all practical purposes, as close to zero as desired. Even with watermarks as small as 64 bits, the probability of random appearance is of the order of 10^{-20}. Hence, the only parameters of interest to the analysis of the statistical performance of the detection scheme are the end-do-end probabilities of packet loss (p_{plo} and p_{pli}) of the server-sniffer channel. The relevant probabilities are, thus, easily obtained:

- **Probability of miss** - p_M. An error of the second kind (failing to detect the watermark) may result both from the loss of the c requests transmitted by the client, or from the loss of all the packets transmitted by the echo-server (n, for each received request). Assuming statistical independence between packets (a waiver to all cases such as jamming or long interference bursts), the conditional probability that the n replies are not detected, given that a request was received by the echo-server, is given by:

$$p_{M|r} = p_{pli}^{n}. \tag{1}$$

  The overall probability of a false negative is given by:

$$p_M = \sum_{i=0}^{c} \binom{c}{i} (p_{plo})^{(c-i)} (1 - p_{plo})^{i} (p_{M|r})^{i}. \tag{2}$$

- **Probability of detection** - p_D. The probability of correct detection of the watermark is given by: $p_D = 1 - p_M$
- **Probability of false alarm** - p_{FA}. An error of the first kind (spurious detection of a watermark) can be considered negligible, as previously discussed, since even small watermarks will bring this probability to negligible values: $p_{FA} \approx 0$.

As can be seen from (1) and (2), increasing n will imply a decrease in the probability of false negatives (p_M). However, there will be a residual probability of false negatives which no increase in n can affect, since it is associated with the probability of loss of all the request packets (the term for $i = 0$, in (2)), a situation where n plays no role whatsoever. This implies that, for a desired probability of detection p_D^*, and even though a trade-off between c and n is possible in general, there is a minimum threshold for c, which can not be compensated by an increase in n:

$$c \geq \frac{log(1 - p_D^*)}{log(p_{plo})} \tag{3}$$

The observed particular values of the parameters (and the resulting probabilities) will be presented at a later section, when discussing the implementation made by the authors. Also, even though we are required to obey (3), there is a trade-off to be made between c and n, since both parameters will affect p_D. However, their effect on the time to complete the test is substantially different: increasing c will imply considerably higher accrued delays. Hence, in our implementation, we kept c as small as possible, and adjusted n to achieve the desired p_D.

4.2 Covert WiFiHop

If the wireless link between the evil and legitimate APs is encrypted, we cannot hope to access the payloads of the exchanged packets. In this case, we modify our scheme, focusing its principle of operation on the one measure we can rely on: packet length. Since the relevant security mechanisms (e.g. WEP, WPA) have deterministic, predictable behaviours, concerning the increase in length of the unencrypted packets [3], we can create an effective watermark using a sequence of packets with pre-determined lengths. Detecting the watermark will then become a problem of detecting a set of packets with the appropriate length sequence. This mechanism also works seamlessly in networks without encryption, being, however, slightly more complex to implement.

Another option, which will not be presented in this article, is to use, as a watermark, not a sequence of chosen lengths, but a sequence of length differentials. This will make the scheme invariant to all network cyphers whose final lengths are affine functions of the plain packet lengths, but will both increase complexity, and decrease the overall statistical performance of Covert WiFiHop. The increase in complexity results directly from the fact that, since we will be operating on a packet switched network, extraneous packets may, and will, be inserted between the watermark packets (more on this will be said later). This creates a hard setup for the detection of length differentials, and will imply an increased scheme complexity. Also, any particular length difference is, statistically (all other things being equal), orders of magnitude more probable than a particular length (many candidate pairs may create that length difference, while length coincidence has a single candidate (that particular length itself). This approach should only be used, therefore, on a *need to* basis, and will not be addressed here.

Before pursuing the analysis of Covert WiFiHop, some notes must be made. The first one concerns the fact that, while in Open WiFiHop, p_{FA} could be considered null, due to the statistical improbability of a random generation of the watermark, in this case, the probability of appearance of extraneous packets with coincidental lengths is not infinitesimal, and such events must, therefore, be considered. In fact, even though with low probability, the appearance of coincidental lengths are not rare events, and its probability depends on the amount of time spent in the analysis of network traffic to support the choice of the least observed packet lengths. To illustrate this, we processed a 4 day sequence of packets from the widely used SIGCOMM conference trace [19] with 10 different analysis periods (0.1 s to 1 s, in 0.1 increments). For each analysis period, we computed the probability that the least observed packet length during the analysis period (or, if the set of least observed lengths is multi-valued, one randomly chosen length from that set) was observed in the 5 minutes immediately following the analysis period. The obtained results can be seen in Figure 3. As can be observed, longer

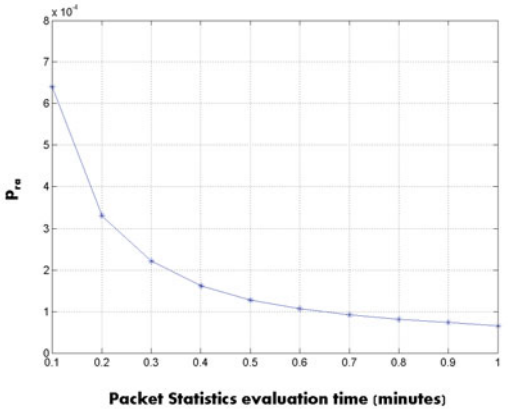

Fig. 3. Probability of extraneous appearance of packets with the chosen length

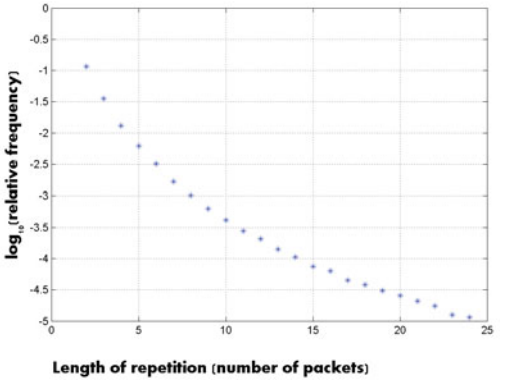

Fig. 4. Probability of sequences of repeated lengths

analysis periods will provide length choices less prone to extraneous appearance. Also, we note that small observations periods (e.g. 6 s) are already very effective in providing low values of p_{ra}.

Secondly, we must consider the fact that common life sequences of packet lengths do not constitute white noise processes. That is, there are significant correlations between successive lengths. Namely, repeating lengths are very frequent. As an illustrative example, we evaluated the number of repetitions in the time series of observed packet lengths from the SIGCOMM trace [19]. To avoid bias in the results, the sequence was preprocessed, and all retransmissions and non-data packets (e.g. beacons) removed. It was found that, in that particular sequence (approximately 29.3 million packets), 17.8% of the packets were immediately followed by one or more packets of the same length. As can be seen in Figure 4, sequences of up to 25 packets with the same length fall in the range of probability $p \geq 10^{-5}$.

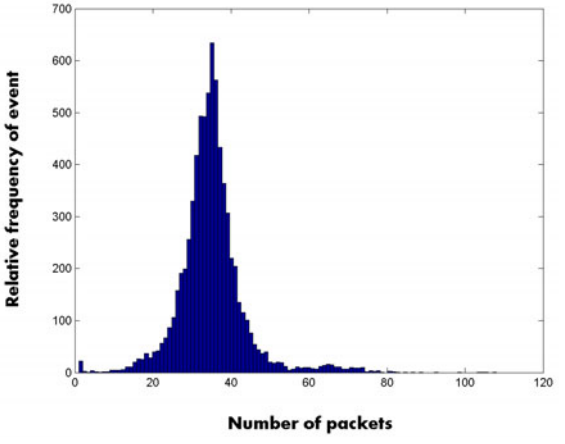

Fig. 5. Distribution of the number of intercalated packets

For confirmation purposes, we made several on campus recordings of 2-hour periods, in different operational situations (weekends and weekdays, morning and late afternoon). The obtained results were, in fact, similar, with the appearance of the same behaviour of repetitions.

This fact immediately rules out the hypothesis of choosing, as watermark, a sequence of packets with the same length, at least for small k. Even if that particular length has a low probability of random appearance, its conditional probability, given that one packet of the coincidental length appeared, is too high. Which, of course, means that the probability of random occurrence of the k-sized sequence may remain of the approximate order of magnitude of the probability of random occurrence of its first value (p_{ra}). We will therefore use as watermark a sequence of packets of different lengths, these lengths being chosen from the set of packets lengths with lower probability (resulting from a local ad-hoc pre-analysis). Since, typically, there will be many lengths of equivalent low probability to choose from, the choice does not pose any additional difficulties or constraints.

Thirdly, we must also consider the fact that, in a packet switching environment, it is possible that extraneous packets are inserted amongst the watermark sequence packets. In fact, for any reasonable traffic load conditions, it will be highly unlikely for the watermark sequence of packets to be transmitted from the legitimate to the evil AP end-to-start, without intercalated extraneous packets, belonging to other conversations. To appreciate this, we can see, in Figure 5, the histogram of the number of packets received in between packets of a 30 packet watermark sequence. Both the details of the test wireless network, and of the traffic conditions in all three profiles (low, medium and high traffic), are detailed in Section 5, more particularly, in Figure 6 and Table 7(a).

The observed mean number of extraneous packets inserted between each pair of packets belonging to the watermark sequence was 34.8. As can be seen in Figure 5, the distribution has a heavy right tail, and inserted sequences of more than 100 packets were observed. The situation becomes worse, of course, at high traffic levels (see

Section 5 for details), were the observed mean rose to 96.4, with the appearance of inserted sequences longer than 300 packets.

Finally, it is also possible for packets in the sequence to arrive at the legitimate AP in reversed order. This is thus the environment which the sequence detection scheme must be designed to cope with.

Detection of the watermark sequence will be made by progressing through the steps of a k-state finite state machine. The machine state progresses whenever a packet with the proper length is detected. If, for example, the watermark is chosen to be a sequence of three packets of lengths 110, 250 and 130, the three-state machine ($k = 3$) will terminate when all three lengths 110, 250, 130 are observed, in this relative order. It does not matter how many extraneous packets are interspersed between the water mark sequence, since the machine state never regresses. Due to the possibility of packet loss, we again have clients repeating the request c times, and the server repeating the watermark n times. That is, n sets of k packets will be transmitted for each request received by the server, each one of the k sets of packets being constituted by packets with the chosen lengths.

Even though the watermark to be detected is constituted by a sequence, we will not use a Sequential Likelihood Ratio Test - SLRT [22] to support the decision, as could be expected in such a type of sequence based detection (see, for example, [28]). The reason is threefold, and lies deep in the assumptions of such a test, which our particular setup fails to obey. To start with, we have the lack of statistical independence between samples of the stochastic process formed by the sequence of packet lengths (mainly resulting from the high probability of repetitions of the same length, as discussed above). But there are two eventually deeper and more structural reasons: the lack of stationarity and lack of ergodicity of the underlying stochastic process (both of which are in the basis of the SLRT). To appreciate this, let us consider a simple example, where an SLRT is being used in a medium traffic situation, and we have just received a packet with the correct length, but we have not yet crossed one of the decision thresholds. This means that we will look at the following packets to further refine the associated probabilities and, hence, hopefully progress towards one of the decision boundaries (see, for example, [15]). The problem is now: is the probabilistic density of the next packet unchanged by the knowledge that we've just detected one packet of the proper length? And what about the density of, say, the 35^{th} packet to arrive after the just detected packet? Is its probabilistic density unchanged? And is it the same density of the next packet? The answer to all these questions is no (which, of course, implies that there is no stationarity, and, hence, no ergodicity). Since, in these traffic conditions, the distribution of the number of packets inserted between watermark packets has a mean of approximately 35 (see Figure 5 and the related discussion), the 35^{th} packet is much more likely to belong to the watermark than the next packet. Those two packets will, therefore, have different associated probability distributions, which immediately rules out stationarity. The assumption of ergodicity cannot thus be maintained (meaning that we cannot reflect the temporal behaviour of the sequence on the statistics of each packet). All in all, the statistical basis for the use of an SLRT is completely compromised.

The chosen decision rule is therefore, the following: if the machine terminates (reaches its final state) within the time period allocated to the test, the watermark is

considered detected (and, therefore, the service providing AP classified as an evil twin); otherwise, it is decided that no watermark was transmitted on that channel, and the test is repeated for the remaining channels.

Statistical Performance. In this case of encrypted networks, the major difference will be observed on p_{FA}, since, as discussed, the probability of extraneous packets having the same length than the marking packets (p_{ra}) is not a negligible quantity. n sets of k packets will be sent, each set having the chosen packet length sequence. Detection occurs when the sniffer has seen all the required packet lengths, in the proper sequence.

We will assume that a false negative due to the loss of the watermark sequence will happen if packet losses occur in all transmitted sets. In fact, this is a conservative assumption, since a partial length sequence, observed on, say, the first set, may be completed by length observations belonging to the second set. Also, it is possible that a lost packet may be substituted by an extraneous packet with the proper length, thus leading to a correct detection of the sequence on that set.

For small values of n, the assumption error is small. In any case, it always leads to a conservative evaluation. With this assumption, the associated probabilities thus become:

- **Probability of miss** - p_M.

$$p_{M|r} = (1 - (1 - p_{pli})^k)^n. \tag{4}$$

  The overall probability of a false negative is, again, given by:

$$p_M = \sum_{i=0}^{c} \binom{c}{i} (p_{plo})^{(c-i)} (1 - p_{plo})^i (p_{M|r})^i. \tag{5}$$

- **Probability of detection** - p_D. The probability of correctly detecting the length sequence is given by: $p_D \geq 1 - p_M$.
- **Probability of false alarm** - p_{FA}. An error of the first kind (spurious detection of the watermark) will occur if we have k extraneous occurrences of appropriate length. Designating by λ the rate of such an extraneous occurrence, we may use a Poisson process to majorate the probability of false alarm:

$$p_{FA} = 1 - \sum_{i=0}^{(k-1)} \frac{(\lambda\, t)^i}{i!} e^{-\lambda\, t}, \tag{6}$$

  where t is the observation period.

This implies that both p_{FA} and P_D can be independently controlled. Increasing k will decrease p_{FA} as desired. This will also reduce p_D, but p_D can then be brought to the desired level by properly choosing n or c. Denoting the desired probabilities by p^*_{FA} and p^*_D, and noting that the previously threshold effect for c is also applicable for the Covert WiFiHop case, we must choose k in such a way as to guarantee that

$$\sum_{i=0}^{(k-1)} \frac{(\lambda\, t)^i}{i!} e^{-\lambda\, t} \geq 1 - p^*_{FA}, \tag{7}$$

and

$$c \geq \frac{log(1 - p_D^*)}{log(p_{plo})} \tag{8}$$

The observed particular values of the parameters, and a description of our particular implementation of WiFiHop, will be presented next. We again note that, in the trade-off to be made between c and n, we kept c as small as possible, to minimize execution time.

5 Implementation

To evaluate the performance of the proposed schemes in a real life situation, a test setup was put together. We deployed an access point in our university campus, and configured it to provide access to the internet. The objective was to test both algorithms under real traffic and environmental conditions, by having several other wireless networks coexisting with our own. An *evil twin* AP was also deployed and configured to act as a rogue access point. Finally, we set up a client with wifihop-ng, and an extra computer that served as a wireless monitor and captured all the wireless packets being sent, for later examination. A representation of this network is shown in Figure 6.

Both the legitimate and the evil twin APs are FON2201, supporting IEEE 802.11b/g. The client used was an Intel desktop, running a Linux flavoured operating system, with a Proxim ComboCard Gold that supports IEEE 802.11a/b/g. The monitor was an iMac running OSX Snow Leopard. Note that all of these off-the-shelf equipment supports monitor mode, and therefore, meets the requirements to run WiFiHop. We tested several different network configurations. In all the tests, the client accessed the evil twin AP through an open network[1] (no encryption), but we varied the encryption available between the evil twin and the legitimate APs. The encryptions used were: WEP with a 128 bit key; WPA2 using AES; and OpenVPN using blowfish. We note here a limitation of our system: the inability of currently dealing with security systems that use random padding on their cryptography algorithm. However, we note that the IEEE 802.11 standard does not offer this feature [3], and the use of VPNs in Hotspots is very uncommon. This leaves WiFiHop capable of detecting the very large majority of present day attacks, as claimed.

To be able to infer the influence of network traffic on the performance of WiFiHop, we added two extra clients to the network, connected directly to the legitimate AP, that generated constant TCP and UDP traffic. The generation of traffic was done using HTTP downloads with rate-limiting, and iperf, a traffic generation tool, used to generate constant UDP traffic streams. The traffic profiles generated by these two extra clients are presented in Table 7(a).

The first task was the characterisation of the parameters relevant to the statistical configuration of the tests. Namely, the following probabilities had to be estimated: the probabilities of packet loss p_{pli} and p_{plo}, the rate of extraneous appearance of packets with lengths coincidental with the length of the next expected watermark packet λ, and the statistical distribution of the roundtrip delay in the user/echo-server channel.

[1] The use of an open network between clients and the evil twin AP is just a particularity of our testing environment, and not a limitation of WiFiHop itself.

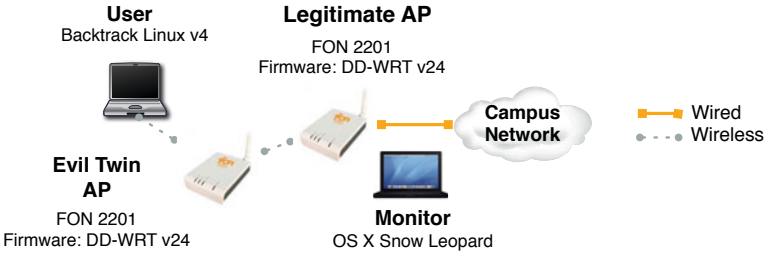

Fig. 6. Illustration of the network in which WifiHop was tested

Profile	DL rate (Mbps)	UL rate (Mbps)
Low	2	1
Medium	8	5
High	16	12

(a) Traffic generated by the extra clients.

	Low Traffic	Medium Traffic	High Traffic
p_{pli} (tcpdump)	0.0217	0.0239	0.0805
p_{pli} (scapy)	0.2574	0.6721	0.8911
p_{plo}	$1.333 \cdot 10^{-4}$	$2.667 \cdot 10^{-4}$	0.16

(b) Probability of packet loss.

Fig. 7. Traffic profiles and probability of packet loss

Determination of λ. We used a six second window to determine the least probable packet lengths. That is: the choice of packet lengths to be used in the watermark sequence (encrypted networks case) is obtained by monitoring network traffic during six seconds, and choosing the k least observed lengths. If there are more than k lengths with the same minimal length, which turns out to be invariably the case, the choice is randomly made between the minimal length candidates. From Figure 3, we see that this choice implies $p_{ra} = 6.4E-4$. Since this probability was obtained for 5 minute blocks, with a mean number of 25606 packets, this translates, in terms of Poisson arrival rate to $\lambda = 0.0533$ arrivals per second.

Determination of p_{pli} *and* p_{plo}. The probabilities of packet loss were experimentally determined. Runs of 10 mins were made, in which control packets were inserted in the network, along with the random traffic generated by the two extra clients (see Table 7(a)). To measure the p_{pli} we divided the number of control packets captured by the sniffers, and divided them by the number of control packets sent by the remote server, effectively obtaining $1-p_{pli}$. For p_{plo}, we simply divided the number of control packets sent from the client, by the number of control packets received at the server, obtaining $1-p_{plo}$. In the case of p_{pli}, and since the sniffer will be the predominant cause of packet loss, we repeated the test with two different sniffers: TCPdump [5] and Scapy [4].

As can be seen in Table 7(b), there is a very big difference in performance between TCPdump and Scapy. Therefore, scapy was used in all the implementation tests, to obtain worst case evaluations.

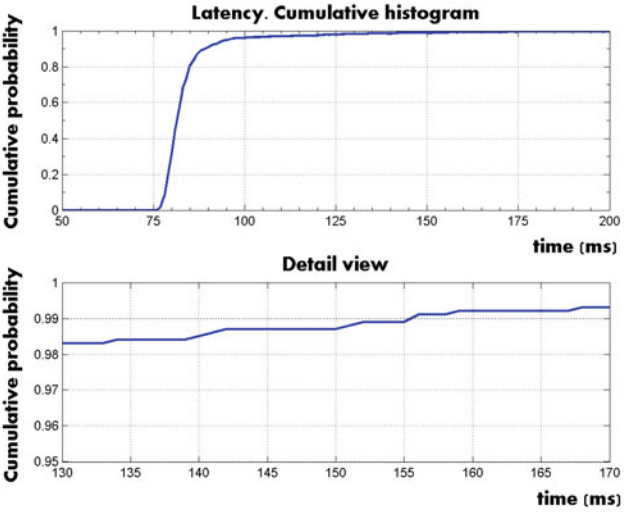

Fig. 8. Cumulative histogram of the round-trip delay

		Low Traffic	Medium Traffic	High Traffic
Open	c	1	1	3
	n	6	19	18
Covert	c	1	1	3
	n	6	19	18
	k	1	1	1

(a) Required values for c, n and k.

Traffic Profile	WiFiHop	Attacks detected
Low Traffic	Open	100%
	Covert	100%
Medium Traffic	Open	100%
	Covert	100%
High Traffic	Open	98.44%
	Covert	98.05%

(b) Effectiveness of WifiHop.

Fig. 9. Parameters and results of WiFiHop's evaluation

Round-trip delay distribution. To evaluate the round-trip delay distribution, a simple experiment was setup: a request was transmitted from the wireless client to the echo-server; upon reception, the echo server replied with a sequence of 30 packets, addressed to the wireless client. The time between the wireless transmission of the request and the wireless reception of the last packet in the sequence was recorded. The experiment was repeated 1000 times, in medium traffic conditions. The cumulative histogram of the recorded values can be seen in Figure 8.

As can be seen in this figure, we only need 156 ms to ensure a 0.99 probability that all the sequence is received. However, the user needs some additional time to be able to change channels, and enter monitor mode. Hence, the echo-server will be made to delay transmissions by an extra 500 ms. This means that the attempt to detect the watermark transmission should not terminate until $t \geq 656$ ms after the transmission of the request (or watermark) to the echo-server. In all tests, t was set to 1 s.

Determination of c, n and k. With the above values of λ, p_{plo} and p_{pli}, it is easy to obtain the needed values of c, n and k, by using the formulas of Section 4. Imposing, as objectives, $p_D^* \geq 0.999$ in the medium and low traffic conditions, $p_D^* \geq 0.98$ in the high traffic case (due to the extremely heavy conditions of the test, well above the limits of reasonable operation), and $p_{FA}^* \leq 0.001$ in all cases, the required values of c, n and k are, for both (Open and Covert) cases, the ones represented in Table 9(a). These were, therefore, the parameter values used in our implementation of WiFiHop, for the performance tests of Section 6.

Since, in the Covert WiFiHop case, k turned out to be one in all traffic conditions (due to the low probability of random appearance of coincidental lengths), the values of c and n are the same for both Open and Covert WiFiHop.

Echo server. The echo-server can deployed through the use of a simple script on any public hosting service (we used a 10-line Python script). This simplicity, opposed to, for example, the deployment and configuration of an end-to-end VPN, enables users to easily create private echo-servers in which they can trust. In Open WiFiHop, the server, after receiving a UDP packet, transmits n replies, each one of them containing the watermark, but delaying the replies by d seconds. In our experiments, clients were always able to switch channels in less than $500ms$. Therefore, $d = 0.5$ was used. The Covert WiFiHop receives, as payload of the UDP request packet, the set of packet sizes to be used in the watermark sequence. The packet sizes, as mentioned in the previous section, are chosen by the user, after testing the wireless medium for six seconds. As in the previous case, the server waits d seconds before transmitting the watermark.

WiFiHop. A command-line tool was developed, which implements both Open and Covert WiFiHop: wifihop-ng. When ran, wifihop-ng puts the wireless device in monitor mode, allowing the wireless device to receive every transmitted packet, regardless of origin or destination. While not all wireless devices support this specific mode, an increasing number of them do [7].

After association with the AP, wifihop-ng verifies internet connectivity by sending a heartbeat to our remote echo-server. Then, it sends the watermark to the AP and immediately switches to one of the other available radio channels, listening all transmitted packets. In Open WiFiHop, we chose a small UDP packet, with a total of 44 bytes, that contains a random 128 bit string in the payload. The choice of UDP over TCP was made to avoid the TCP three-way handshake. However, any kind of packet can be used. In Covert WiFiHop, the watermark is also UDP packet, containing the packet sizes to be used on the watermark (see Section 4 for more details on this).

To distinguish between both mechanisms wifihop-ng receives an input flag. If the watermark is not detected within the time period t, the procedure is repeated with a different wireless channel. Conversely, if the watermark is detected, wifihop-ng returns the origin and destination MAC addresses of the APs involved in the communication. The user is then asked if he wishes to connect directly to the legitimate AP.

6 Results

Regarding the effectiveness of WiFiHop, we show in Table 9(b) the results of a total of 6000 different trials, using the previously calculated parameters (Table 9(a)). We made 1000 tests for each one of the traffic profiles present in Table 7(a), for both Open and Covert WiFiHop. The immediate conclusion is that WifiHop had no false negatives for both the low and medium traffic profiles. This was expected, since we had imposed $p_D \geq 0.999$ (and, hence, $p_M \leq 0.0001$) in both these settings. In the high traffic case, the observed detection probabilities were $p_D = 0.9844$ and $p_D = 0.9805$ (for Open and Covert WiFiHop, respectively), which again are within the required range. In none of the tests false positives occurred. That is why this parameter is not shown in Table 9(b).

The user does not need to test every available wireless channel, since wifihop-ng can search for beacons, and tests can safely be made only on channels in which networks are being advertised. However, we will consider the worst case scenario (testing all 11 channels). Each test requires six seconds to choose the less probable packet lengths (in the case of Covert WiFiHop), an additional $t = 1$ seconds for each wireless channel to be tested, and approximately 0.5 seconds to change the channel back to the network operation channel. This means that a full set of tests will last a minimum of approximately 22.5 seconds. In fact, all the tests performed with wifihop-ng, required approximately 30 seconds. Such a full set of tests will only be done once, when the user joins a new network, and therefore, this time-frame does not seem to be excessive or impractical.

7 Conclusions

User-sided evil twin attack detection is viable. It can be done in useful time, and is statistically high effective in the range of normal network operations. These detection mechanisms can operate in both open and encrypted networks (e.g. WEP, WPA and some VPNs). Also, they avoid many of the difficulties associated with some server-sided detection mechanisms, such as the need for several wireless sniffers, the high false positive rate, and the non-real time detection. We have shown that WiFiHop can be implemented in off-the-shelf equipment, giving wireless hotspot users the capability of individually detecting a evil twin attack in the networks to which they access, without having to trust the network operator. In our implementation of WiFiHop, this detection was always done in less than one minute, with virtually no false positives, and a very low rate of false negatives.

References

1. Airdefense - tire of rogues? solutions for detecting and eliminating rogue wireless networks, `http://www.airdefense.net/whitepapers/roguewatch_request2.php`
2. Netstumbler, `http://www.netstumbler.com/`
3. Nist guide to securing legacy ieee 802.11 wireless networks, `http://csrc.nist.gov/publications/nistpubs/800-48-rev1/SP800-48r1.pdf`
4. Scapy project, `http://www.secdev.org/projects/scapy/`
5. Tcpdump, `http://www.tcpdump.org/`

6. Wavelink, http://www.wavelink.com
7. Wireless card compatibility list, http://www.aircrack-ng.org/doku.php?id=compatibility_drivers
8. Wisentry - wireless access point detection system, http://www.wimetrics.com/Products/WAPD.htm
9. Abdollah, T.: Ensnared on the wireless web, http://articles.latimes.com/2007/mar/16/local/me-wifihack16
10. Adya, A., Bahl, P., Chandra, R., Qiu, L.: Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, MobiCom 2004, pp. 30–44. ACM, New York (2004), http://doi.acm.org/10.1145/1023720.1023724
11. Bahl, P., Chandra, R., Padhye, J., Ravindranath, L., Singh, M., Wolman, A., Zill, B.: Enhancing the security of corporate wi-fi networks using dair. In: Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, MobiSys 2006, pp. 1–14. ACM, New York (2006), http://doi.acm.org/10.1145/1134680.1134682
12. Baiamonte, V., Papagiannaki, K., Iannaccone, G.: Detecting 802.11 wireless hosts from remote passive observations. In: Akyildiz, I.F., Sivakumar, R., Ekici, E., Oliveira, J.C.d., McNair, J. (eds.) NETWORKING 2007. LNCS, vol. 4479, pp. 356–367. Springer, Heidelberg (2007), http://portal.acm.org/citation.cfm?id=1772322.1772361
13. Bellardo, J., Savage, S.: 802.11 denial-of-service attacks: real vulnerabilities and practical solutions. In: Proceedings of the 12th Conference on USENIX Security Symposium, vol. 12, p. 2. USENIX Association, Berkeley (2003), http://portal.acm.org/citation.cfm?id=1251353.1251355
14. Beyah, R., Kangude, S., Yu, G., Strickland, B., Copeland, J.: Rogue access point detection using temporal traffic characteristics. In: Global Telecommunications Conference, GLOBECOM 2004, November-December 3, vol. 4, pp. 2271–2275. IEEE, Los Alamitos (2004)
15. Hippenstiel, R.D.: Detection Theory: Applications and Digital Signal Processing, 2nd edn. CRC Press, Boca Raton (2002)
16. Kao, K.F., Liao, I.E., Li, Y.C.: Detecting rogue access points using client-side bottleneck bandwidth analysis. Computers and Security 28(3-4), 144–152 (2009), http://www.sciencedirect.com/science/article/B6V8G-4V353XY-1/2/0e2cd909933fa11ae60a0417d16d0faa
17. Ma, L., Teymorian, A.Y., Cheng, X.: A Hybrid Rogue Access Point Protection Framework for Commodity Wi-Fi Networks. In: 2008 IEEE INFOCOM - The 27th Conference on Computer Communications, pp. 1220–1228. IEEE, Los Alamitos (2008), http://dx.doi.org/10.1109/INFOCOM.2008.178
18. Mano, C.D., Blaich, A., Liao, Q., Jiang, Y., Cieslak, D.A., Salyers, D.C., Striegel, A.: Ripps: Rogue identifying packet payload slicer detecting unauthorized wireless hosts through network traffic conditioning. ACM Trans. Inf. Syst. Secur. 11, 2:1–2:23 (2008), http://doi.acm.org/10.1145/1330332.1330334
19. Schulman, A., Levin, D., Spring, N.: CRAWDAD data set umd/sigcomm2008 (March 2, 2009), crawdad.cs.dartmouth.edu/umd/sigcomm2008 (March 2009)
20. Shetty, S., Song, M., Ma, L.: Rogue access point detection by analyzing network traffic characteristics. In: Military Communications Conference, MILCOM 2007, pp. 1–7. IEEE, Los Alamitos (2007)
21. Song, Y., Yang, C., Gu, G.: Who is peeping at your passwords at starbucks?; to catch an evil twin access point. In: 2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), June 28- July 1, pp. 323–332 (2010)
22. Wald, A.: Sequential Analysis. Wiley, Chichester (1959)

23. Watkins, L., Beyah, R., Corbett, C.: A passive approach to rogue access point detection. In: Global Telecommunications Conference, GLOBECOM 2007, pp. 355–360. IEEE, Los Alamitos (2007)
24. Wei, W., Wang, B., Zhang, C., Kurose, J., Towsley, D.: Classification of access network types: Ethernet wireless lan, adsl, cable modem or dialup? In: Proceedings IEEE of INFOCOM 2005 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2, pp. 1060–1071 (March 2005)
25. Wei, W., Jaiswal, S., Kurose, J., Towsley, D.: Identifying 802.11 traffic from passive measurements using iterative bayesian inference. In: Proc. IEEE INFOCOM (2006)
26. Wei, W., Suh, K., Wang, B., Gu, Y., Kurose, J., Towsley, D.: Passive online rogue access point detection using sequential hypothesis testing with tcp ack-pairs. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC 2007, pp. 365–378. ACM, New York (2007), http://doi.acm.org/10.1145/1298306.1298357
27. Xie, G., He, T., Zhang, G.: Rogue access point detection using segmental tcp jitter. In: Proceeding of the 17th International Conference on World Wide Web, WWW 2008, pp. 1249–1250. ACM, New York (2008), http://doi.acm.org/10.1145/1367497.1367750
28. Yin, H., Chen, G., Wang, J.: Detecting protected layer-3 rogue aps. In: Fourth International Conference on Broadband Communications, Networks and Systems, BROADNETS 2007, pp. 449–458 (September 2007)