A Measurement Based Rogue AP Detection Scheme

Hao Han[†‡], Bo Sheng[‡], Chiu C. Tan[‡], Qun Li[‡], Sanglu Lu[†]
[†]State Key Laboratory of Novel Software Technology, Nanjing University, China
[‡]College of William and Mary, Williamsburg, VA, USA
Email: [†]hanhao@dislab.nju.edu.cn, [‡]{shengbo, cct, liqun}@cs.wm.edu, [†]sanglu@nju.edu.cn

***Abstract*—This paper considers a category of rogue access points (APs) that pretend to be legitimate APs to lure users to connect to them. We propose a practical timing based technique that allows the user to avoid connecting to rogue APs. Our method employs the round trip time between the user and the DNS server to independently determine whether an AP is legitimate or not without assistance from the WLAN operator. We implemented our detection technique on commercially available wireless cards to evaluate their performance.**

I. Introduction

The use of IEEE 802.11 based wireless local area networks, or WLANs, has grown in popularity in recent years. As people's expectation of ubiquitous wireless availability increases, the security of such networks becomes more important. The problem of *rogue access points* has emerged as an important security problem in WLANs, as demonstrated by commercial products [1], [2], [3] and academic research [4], [5], [6], [7] devoted to this problem.

A rogue access point, or rogue AP, is an access point that is not deployed by the WLAN administrator. A rogue AP can be set up by connecting an AP directly into an Ethernet jack on a wall, or by using two wireless interfaces. The first interface is connected to a real AP, and the other acts as an AP to lure users. The differences between the two will be examined later in the paper. In this paper, the term "rogue AP" refers to a rogue AP that utilizes two wireless interfaces. In this scenario, once a user is connected to a rogue AP, the adversary can manipulate all data sent and received by the user, allowing the adversary to launch different kinds of attacks.

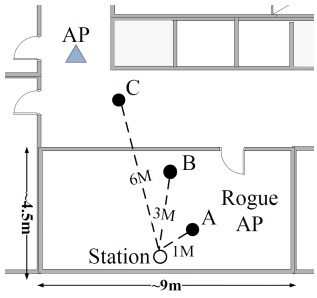

Fig. 1. Illustration of Rogue AP Attacks: In our experiments, a rogue AP is deployed at location A, B and C.

To demonstrate the effectiveness of our rouge AP, we set up the testbed shown in Fig. 1. The station laptop is placed in an office several meters away from a legitimate AP mounted on the ceiling. The SINR of legitimate AP measured by the station is 40dbm. We then setup the rogue AP and place it at three separate locations A,B, and C. Location A is one meter away from the station, location B is three meters away, and location C is 6 meters away behind a wall. The goal is to determine if we could induce the station to connect to the rogue AP instead of the legitimate AP.

Table I shows the SINR values received by the station when the rogue is placed at different locations. By default, the station will select the AP with the highest SINR to connect to. In our experiments, when the rogue AP's SINR is greater than 40 dbm, it is highly likely that the station will be lured into connecting with the rogue AP.

TABLE I
Average SINR under different distance and TX power

TX Power (dbm)	SINR (dbm)		
	A (1m)	B (3m)	C (6m through a wall)
18 (default)	71	55	40
14	67	51	36
10	61	47	33

In this paper, we propose a rogue AP detection technique that allows the user to independently determine whether an AP is a legitimate or not without assistance from the WLAN operator. To the best of our knowledge, we are the first to propose a rogue AP detection scheme that can be implemented purely by end users. Our main contributions are: (1) We propose a timing based rogue AP detection algorithm that relies only on existing networking protocols to work, and can be applied to any regular WLAN network without requiring further modifications by network administrators. (2) We consider a more powerful *malicious* adversary that actively manages the rogue AP to avoid detection as opposed than an "accidental" rogue AP deployed, for example, by an innocent employee in an office [8]. (3) We implement our scheme using commercial off-the-shelf wireless cards and evaluate the performance through real experiments.

The rest of the paper is organized as follows. Sections II and III discuss the related work, and problem formulation respectively. Our algorithm is detailed in Section IV, and our implementation is presented in Section V. We depict the evaluation results in Section VI and conclude in Section VII.

II. Related Work

The threat of rogue APs have attracted the attention of both commercial and academic researchers. One approach is to deploy wireless sniffers to monitor wireless traffic. These

sniffers collect wireless traffic and achieves the data for further analysis. Work by [9], [8], [10] collected the MAC addresses from the wireless traffic and compares them against a database of known MAC addresses. Unknown MAC addresses indicate the presence of rogue APs. Since a malicious rouge AP can spoof its MAC address, other researchers have focused on using sniffers to collect RSS values [11], clock skews [12], and radio frequency variations [13] to identify rogue APs. Our solution differs from these work in that we do not require installing and maintaining expensive sniffers.

Another approach analyzes network traffic to detect the presence of rogue APs [4], [5], [6]. Characteristics such as the spacing between packets are different for a wired and wireless network [14], [7]. Wireless standards such as CSMA/CA, and physical properties like half duplex channels also help distinguish wireless traffic from wired traffic [5], [6]. The presence of wireless traffic at a particular gateway where there should be only wired traffic will indicate the presence of a rogue AP. A common requirement is to monitor traffic for the entire network. This is possible only in environments such as corporate networks where cooperation with network provider can be assumed. While our algorithm also relies on unique wireless characteristics, the traffic traces we use are self generated. Our solution does not require the cooperation of network service provider.

III. Problem Formulation

A. Problem Definition

We consider a scenario that a wireless station tries to join a WLAN to access the Internet. After scanning the channels, the station will discover multiple APs within its communication range. Some of these APs are legitimate and some might be rogue APs. Our objective is to design an algorithm that helps the station detect the rogue AP. The detection algorithm should be a complementary part to the existing AP selection policy. Furthermore, the detection algorithm should function in all IEEE 802.11 based wireless networks without further modifications from the network administrator.

We assume that the rogue AP will be launched using a mobile device with two wireless interfaces. The first interface connects the rogue AP to the legitimate AP. The second interface pretends to be a legitimate AP to induce users to connect to it. When a user associates with the rogue AP, the rogue will forward packets from the second interface to the first interface, and then towards the legitimate AP. This way, the user will still be able to access the Internet as if connected to a real AP. Fig. 2 illustrates the setup.

We do not consider a rogue AP setup where the adversary directly plugs the rogue AP into an Ethernet jack in the wall. There are three reasons for this. First, there are a limited number of available Ethernet jacks in public places like airports, making this type of rogue APs less likely in such places. Second, since these rogue APs must remain connected to the Ethernet, they cannot move closer to users to increase their SINR to induce people to connect to them, thus limiting their impact. Finally, network administrators can configure the network to disallow unknown devices from being assigned IP addresses when plugged into the Ethernet network. In this case, the rogue AP will be unable to provide Internet access to the users, making them easy to be detected by the user.

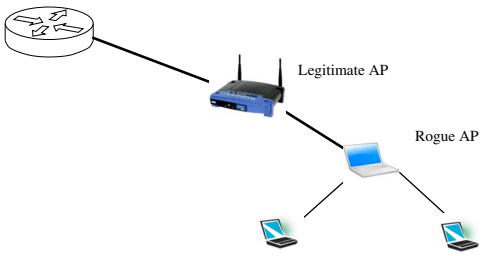

Fig. 2. This figure shows the setup for the rogue AP. A rogue AP is connected to the wired network through a legitimate AP. Some stations inadvertently connect to the rogue AP because the rogue AP is closer and broadcasts stronger signals to them.

B. Malicious Adversary

Here we consider some defenses that can be circumvented by a sophisticated adversary.

1) Identity verification: Users can run programs like `traceroute` to determine whether the connected AP is a rogue AP. `traceroute` will return the number of intermediate hops to a host site. From the output, the station will learn that a suspicions AP exists in the route. However, the rogue AP can evade detection by monitoring the wireless channel to learn the SSID and MAC address of a legitimate AP, and then setup the rogue AP to have the same parameters. The rogue can then avoid forwarding the real AP's reply to the user, thus giving the impression that it is connected to the same gateway as a legitimate AP.
2) Traffic monitoring: Traffic monitoring is a technique to distinguish between wireless and wired traffic. For instance, [5] monitors all the traffic at a gateway and computes the interval between two consecutive TCP ACK packets. A longer interval indicates that the TCP packets are traveling over a wireless connection.
However, this technique cannot distinguish between a legitimate AP and a rogue AP since both utilize a wireless link.
3) Simple timing: The station may use the timing information such as the round trip time (RTT) to detect a rogue AP. Since the rogue AP consists of an additional wireless link to the legitimate AP, this may lead to a delay when transmitting data. The station can determine the RTT by sending a message such as a `ping` request or TCP data packets [14] and waiting for a reply.
However, the rogue AP can simply create a response to return to the user, thus avoiding the time penalty of the additional wireless link. For instance, the rogue AP can generate a `ping` response to return to the user without forwarding the request to the real AP. Similarly when

the user sends a TCP packet, the rogue AP can return the ACK to the user directly.

IV. Our protocol

Our rogue AP detection protocol also uses a timing information based on round trip time (RTT). We begin with examining the motivation and challenges of our approach, followed by some background discussion. We then propose our protocol and examine the parameter values.

A. Motivation and Challenges

There are two reasons for using RTT as a basis for our protocol. First, when a user connects to the network via a rogue AP, all his packets traverse two wireless hops, one between the user and the rouge AP, and the other between the rogue AP and the real AP. When the user is communicating with a real AP, there is only one wireless hop. This additional hop will introduce an unavoidable time latency assuming that the rogue is forced to communicate with the real AP.

We believe the RTT-based method is suitable for rogue AP detection due to the following reasons. First, measuring RTTs is able to distinguish the route through a rogue AP with that through a legitimate AP. Comparing these two cases, the difference is that the route from the user to the server via a rogue AP needs an extra wireless transmission before reaching the wired network. Similarly, the response from the server has to pass one more wireless link through the rogue AP. Considering the channel contention delay and relatively low transmitting rate, the wireless transmission consumes the dominate part in the RTT. Therefore, two extra wireless transmissions can yield a visible difference in the measured RTTs. Second, it is feasible and easy for the user to measure RTTs. Unlike non-timing methods mentioned in related work, measuring RTTs does not need any special equipments, such as sniffers [1], [2] or radio frequency analyzers [15]. It does not require modification at server or AP side.

However, applying RTT-based method is not straightforward. It has to address the following three issues in order to effectively detect the rogue AP. (1) The first issue is which server to contact. A server in the local network is preferred over a remote server in the Internet because the RTT-based method is sensitive to the delay in the wired network. Probing a remote server may cause a significant variance of RTT due to the dynamic routing path and Internet traffic. (2) The second issue is what probing message to use. We want a probing message that cannot be easily manipulated by the rogue AP and can certainly reach the server in any network setting. As we mentioned earlier, a simple ping message can be easily returned by the rogue AP to evade detection and might be blocked by some network administrators. In addition, our probing message has to adhere to the existing networking protocols so as to avoid requiring assistance from the network provider. (3) Finally, we have to consider the effect of network traffic conditions. A busy channel may adversely affect RTT timing and lead to incorrect rogue AP detection.

B. Background

In our solution, we use DNS lookup and answer to solve the first and second issues above, and we use 802.11 management frames, probe request and response in particular, to tackle the third problem.

DNS lookup and answer: The basic function of DNS is to provide a distributed database that maps human-readable host names (such as `www.cs.wm.edu`) to IP addresses (such as `128.239.26.64`). The nodes managing this distributed database are called DNS servers that serve to answer the lookup request for specified host names. To achieve high performance of DNS lookup, caching the queried records is extensively used in current networks.

Typical DNS lookup can be classified into two types: recursive query, and nonrecursive query. In a recursive DNS lookup, a station queries a local server for a host name. If this server cannot answer the query, it will contact the root DNS server which will then recursively ask other servers to determine the IP address. For a nonrecursive query, the local DNS server will only search the cached records locally without contacting the root DNS server. If no match is found, the local server will send a "no such host name" message back to the station.

In our algorithm, we use nonrecursive query as the probing message to measure the RTT between the user and the DNS server. The user will send a DNS lookup request for a host name with the nonrecursive option. The host name may be a known host name or non-existing name. Then the user waits for the response from the local DNS server and measures the RTT by subtracting the transmitting time from the time of receiving the answer. The user repeats this process with a different host name each time.

Our proposed scheme is efficient since most local networks will have a local DNS server or resolver for performance reasons [16]. Therefore, a station can always send a request to the local DNS server and the time spent on the wired network is small due to the local communication. Furthermore, DNS is required by all networks, and thus unlike ping messages, cannot be blocked by network providers. Finally, the DNS response varies for different queries. The adversary cannot predict in advance what the user may choose to query, nor determine whether a particular query can be satisfied by the actual DNS server. The adversary that returns an incorrect reply will be detected by the user. As a result, the adversary must forward the request to the real DNS server to ensure that the reply is correct.

Besides, since the radio range of a station is limited, it is not unreasonable to assume all APs (including rogue APs) detected by the station belong to the same network, and use one common DNS server. The answers of DNS lookup from those APs should be the same. Thus, by comparing results from tested APs, we can suspect the APs that always reply "no such host name" as rogues. In an alternative way, a station can pick an AP to make the DNS server cache some DNS queries beforehand, and then test the remaining APs. This strategy

also forces a rogue AP to forward the request.

Probe request and response: To determine the wireless traffic conditions, we measure another RTT using *probe request* and *probe response* messages. These messages are typically used when a station is scanning for APs to connect.

There are two advantages to use probe request and response. First, by calculating the durations between these two packets, we can estimate the channel traffic and the AP's workload. If the channel is busy, both the probe request and response will take a long time to transmit due to channel contention and retransmission after signal collisions. In addition, if the AP is heavily loaded, i.e., the AP is sending many packets for other associated stations, the probe response message has to wait in the AP's transmission queue for a long time before being sent out. Second, in practice, it is difficult for a smart rogue AP to fake a busy channel condition by intentionally delaying the probe response for a long time, since wireless card driver does not dispatch this kind of low level management frame to OS, and the rogue AP cannot control probe packets without modifying drivers.

The regular probe request, however, is not perfectly suitable for our scheme because it is a broadcast message and will be replied by every AP that hears it. This leads to multiple responses that create unnecessary channel contention and biased RTT measurements. Furthermore, a broadcast message does not have any retransmission. If the associated AP does not receive probe request correctly, the station will not receive the probe response. One possible solution is to apply a timeout mechanism at the higher layer to retransmit the probe request. However, it may incur a large overhead. Thus, we modify the probe request packet to be a unicast message. We put the MAC address of the target AP into the destination field in the probe request so that only the AP in consideration will reply. This also ensures that the target AP will reply MAC layer ACK in case of successful delivery and otherwise the retransmission is enabled.

C. Protocol

In this subsection, we present the overview of our rogue AP detection scheme. We use sta to indicate a station. For a given AP_x within sta's communication range, the station runs the following Algorithm 1 to determine whether AP_x is a rogue AP.

Our algorithm consists of two phases. The first phase (lines 2-5) is to measure the RTTs, and the second phase (lines 6-15) is to analyze the collected RTTs and decide if the current tested AP is rogue. In the first phase, the station repeatedly sends a probe request (line 3) and a DNS lookup (line 4) for n rounds. Respectively, RTT_{probe} (RTT_{dns}) records the round trip time between probe (dns) request and the response. Note that we subtract the data transmission time T_{data} from both RTT_{probe} and RTT_{dns}, because probe packets and DNS packets have different packet sizes and transmission rates, and may vary in each round. After eliminating the effects caused by data transmission time, we can fairly compare RTT_{probe} and RTT_{dns}. The details of how to calculate T_{data} are discussed in next subsection. The number of rounds n is another important parameter capturing the tradeoff between the overhead and accuracy. Larger n incurs a larger overhead, but increases the detection accuracy. According to our experimental observations, $n = 100$ will yield sufficiently accurate results. In the second phase, we first filter out some outlier RTTs (line 6). Due to the page limit, we omit the details of this filtering process. After that, we calculate the mean value (lines 7-8) and standard deviation (lines 9-10) of both RTT_{probe} and RTT_{dns}. Finally, in lines 11-15, we check the difference between these two RTTs Δt (line 11) against a threshold parameter θ (line 12) to determine if this is a rogue AP. The threshold θ which reflects the delay induced by the extra wireless transmissions in rogue AP case, is calculated by σ_{probe} and σ_{dns} according to experimental measurements. We will present the details of function f in the next subsection.

Algorithm 1 Detecting Rogue AP (AP_x)

1: Connect and associate with AP_x
2: **for** $i = 1$ to n **do**
3: Send *unicast* probe request to AP_x, record round trip time $RTT_{probe} = RTT_{probe} - T_{data}(probe)$.
4: Send DNS lookup to local DNS server, record round trip time $RTT_{dns} = RTT_{dns} - T_{data}(dns)$.
5: **end for**
6: Filter outliers
7: $\overline{RTT_{probe}}$ = Mean of remaining RTT_{probe}
8: $\overline{RTT_{dns}}$ = Mean of remaining RTT_{dns}
9: σ_{probe} = Standard deviation of remaining RTT_{probe}
10: σ_{dns} = Standard deviation of remaining RTT_{dns}
11: $\Delta t = \overline{RTT_{dns}} - \overline{RTT_{probe}}$
12: $\theta = f(\sigma_{probe}, \sigma_{dns})$
13: **if** $\Delta t > \theta$ **then**
14: AP_x is a rogue AP
15: **end if**

D. Parameter values

Here, we explain how to derive T_{data} and θ in Algorithm 1. We begin with a quick review of the 802.11 protocol.

IEEE 802.11 medium access control adopts CSMA/CA model and the distributed coordination function (DCF). Before transmitting a frame, a station first senses whether the channel is idle. If the channel is busy, the frame transmission will be deferred until the channel becomes available. After the channel is free for certain period of time, which is defined as DIFS, the station starts a back-off operation with a slot counter whose value is randomly selected between 0 and the size of a contention window (CW_{min}). This back-off counter decreases by one for each idle slot time. When the back-off counter becomes zero, the station transmits the frame. When the destination receives the frame successfully, it sends a MAC layer ACK back to notify the sender. If the sender does not receive an ACK, it will retransmit the packet. Table II lists some timing parameters in 802.11 standard that we will use later.

TABLE II
IEEE 802.11 CHARACTERISTICS

Parameters	Values		
	802.11b	802.11g	802.11a
t_{slot}	20 μs	9 μs	9 μs
t_{DIFS}	50 μs	34 μs	28 μs
t_{PCLP}	192/96 μs	192/96 or 20 μs	20 μs
CW_{min}	31	15	15

Based on 802.11 mechanism, we can express the delay for transmitting a packet as

$$T_{delay} = t_{DIFS} + t_{bf} + t_{defer} + T_{data} + t_{retransmit},$$

where t_{defer} is the time deferred due to the busy channel medium, $t_{retransmit}$ is the time for retransmission if no ACK is received, and t_{bf} is the random back-off time. The expected value of t_{bf} is given by

$$\overline{t_{bf}} = \frac{CW_{min}}{2} \cdot t_{slot}.$$

Data transmission time T_{data} depends on the data size (L-byte payload) and the transmitting rate (r Mbps),

$$T_{data}(L, r) = t_{PCLP} + \frac{(28 + L) \cdot 8\text{bits}}{r},$$

where t_{PCLP} is the physical layer packet overhead of any IEEE 802.11 packet including two parts: the Physical Layer Convergence Protocol (PCLP) preamble used for synchronization and the PCLP header. According to the standard, T_{PCLP} is $192\mu s$ ($96\mu s$) for long (short) preamble, using ERP-DSSS modulation scheme (supporting 1-11 Mbps). And T_{PCLP} is $20\mu s$, using ERP-OFDM modulation scheme (supporting 6-54 Mbps). Value 28 is the length of MAC header plus CRC checksum. For every measured RTT at each round, the station is aware of the data size (L) and transmitting rate (r) for every incoming and outgoing packet. It thus can compute the exact values of $T_{data}(probe)$ and $T_{data}(dns)$, and subtract them from RTTs to eliminate the effect of different transmission time.

In order to derive θ, we need analyze and express the RTTs. For a legitimate AP, the path taken for a probe is

$$STA \rightarrow AP \rightarrow STA,$$

and the path taken for a DNS lookup and answer is

$$STA \rightarrow AP \rightarrow SERV \rightarrow AP \rightarrow STA.$$

For simplicity, we only consider the network overhead, and ignore the time for AP an DNS server to process the packets. There, after subtracting T_{data}, these two RTTs for probe and DNS can be expressed as

$$RTT_{probe} \approx T^{sta \rightarrow ap}_{overhead} + T^{ap \rightarrow sta}_{overhead}$$

and

$$RTT_{dns} \approx T^{sta \rightarrow ap}_{overhead} + T_{wired} + T^{ap \rightarrow sta}_{overhead},$$

where $T_{overhead}$ is used to indicate the remaining part of T_{delay} after deducting T_{data}. Since the RTTs of DNS and probe are measured at approximately the same time, we assume the network conditions are stable during that time period[1], so that we regard $T_{overhead}$ of probe and DNS as the same, and the difference between the two RTTs is

$$\Delta t = \overline{RTT_{dns}} - \overline{RTT_{probe}} = T_{wired}.$$

Considering the efficiency of DNS query, a local DNS server cannot locate too far away from stations in local network. Based on our extensive experimental measurements (which will be shown later in Section VI), Δt is no larger than 1m, even when we consider 5 hops between sending a DNS lookup and receiving the answer back in the idle network traffic condition.

On the other hand, if the tested AP is a rogue AP, the path taken for a probe is still

$$STA \rightarrow RAP \rightarrow STA,$$

but the path for DNS is

$$STA \rightarrow RAP \rightarrow AP \rightarrow SERV \rightarrow AP \rightarrow RAP \rightarrow STA.$$

Similarly, we get

$$\begin{aligned} \Delta t' &= \overline{RTT_{dns}} - \overline{RTT_{probe}} \\ &= T^{rap \rightarrow ap}_{delay} + T_{wired} + T^{ap \rightarrow rap}_{delay}. \end{aligned}$$

For a station, it is difficult to estimate T_{delay} between the rogue AP and its associated legitimate AP, since the station does not know the transmitting rate and network condition at the AP side. However, based on our experiments, we observe that $\Delta t'$ is larger than 1ms in the idle traffic condition, even when the rogue AP transmits with the maximum rate of 54Mbps. For our Algorithm 1 to effectively detect a rogue AP, θ ideally needs to be set between Δt and $\Delta t'$ as $\Delta t < \theta < \Delta t'$. Therefore, we set the threshold θ to 1ms for the idle traffic condition.

Next, we consider the scenarios where the network traffic and AP's workload are heavy. Both Δt for a legitimate AP and $\Delta t'$ for a rouge AP will definitely increase, since the heavy traffic and workload congest the AP causing long tx queues, packet loss, and retransmissions. Thus, the corresponding response time becomes longer. Recall the process of DNS request, when an AP receives a DNS lookup, it will send the request to the DNS server, and wait for the answer. When the traffic is heavy, many packets will be queued in the AP during the waiting time, thus the DNS answer will not be sent by the AP until all packets ahead are transmitted. Especially in the rogue AP case, the additional wireless hop will make $\Delta t'$ much larger. Therefore, the threshold 1ms set in the idle traffic condition is too small to be effective in the heavy traffic condition. In our solution, we dynamically adjust the threshold θ according to the standard deviation of RTT_{probe} and RTT_{dns}. Based on our experiments, we observe that the standard deviation of measured RTTs in the heavy traffic condition is larger than that in low traffic condition. Fig. 3

[1][17] mentions that wireless network traffic remains stable within approximately $150 - 250$ ms in practice.

illustrates the mean value against the average standard deviation of measured RTTs in our extensive experiments under different traffic loads. As we can see, using a line starting

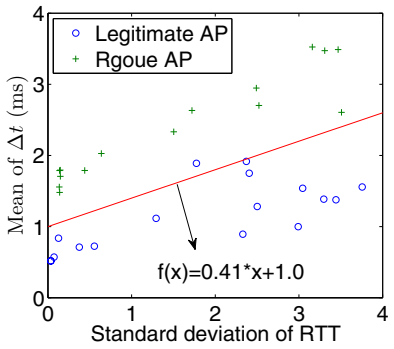

Fig. 3. Mean value of Δt against the average standard deviation of RTT_{probe} and RTT_{dns} under different traffic load condition

from (0,1) (initial threshold 1ms for the idle traffic condition) with slope 0.41, we can distinguish between legitimate AP and rogue AP in most of the tested cases. Therefore, we set

$$\theta = f(\sigma_{probe}, \sigma_{dns}) = 0.41 \cdot \frac{\sigma_{probe} + \sigma_{dns}}{2} + 1.0 \quad (1)$$

Recall in Algorithm 1, after measuring the two RTTs, a station computes the Δt and corresponding θ, according to the mean value and standard deviation of RTT_{dns} and RTT_{probe}. If $\Delta t > \theta$, the station will mark the AP it connects to as a rogue AP. Then the station should choose another AP for test. An alternative way is to compare the difference (Δt) of all the nearby APs, and classify those APs into different categories according to the value of Δt. Rogue APs must belong to a isolated set. We will investigate this approach in our future work.

V. Implementation and Setup

A. Hardware description

Fig. 4 illustrates the infrastructure of our testbed which consists of two APs and three laptops: one laptop is used as a traffic generator, and the remaining laptops serve as the station and rogue AP. Server A is the DNS server in our campus network. To investigate the effect of wired link on our algorithm, we set another DNS server B in the same subnet of APs.

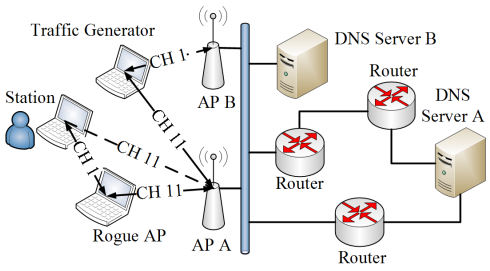

Fig. 4. Illustration of the architecture of the testbed

The hardware specification of each component is described as follows.

(a) Access Points: We used a Linksys WPA54G and D-Link DI-624+A for our APs. Both APs operate in the 802.11g mode.

(b) Wireless Stations: All laptops in Fig. 4 are 2GHz x86 machines running Linux 2.6x kernel. The traffic generator and station are equipped with a TP-Link TL-WN610G wireless card while the rogue AP possesses 2 wireless cards, one TP-Link TL-WN610G, and the other Intel 3495ABG.

(c) DNS server: DNS server B is a desktop computer connected in local wired network running dnsmasq.

B. Software Description

(a) Drivers: We use Madwifi v0.9.4 driver [18] for the wireless cards with Atheros chipset and ipw3945 v1.2.2 driver [19] for those with Intel chipset.

(b) Click toolkit on station: On the station side, we implement the proposed algorithm using Click [20] toolkit with the wireless card turned into promiscuous mode. We inject our codes into the Click modules to record accurate system time (in microseconds) when a probe request (DNS lookup) is pushed to transmit queue of wireless interface, and when the probe response (DNS answer) is received successfully (corresponding interrupt handler is called). We also implement our own unicast probe request. The comparison between two probe requests are shown in Fig 5.

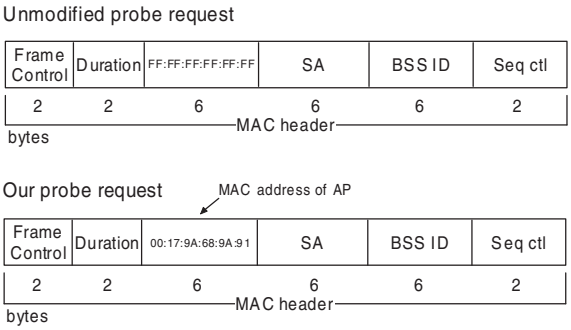

Fig. 5. MAC header comparison between unmodified probe request and our probe request

(c) Iptables and Macchanger on the rogue AP: For the rogue AP, one of its wireless cards is configured to work in the AP mode in order to attract the station, and the other wireless card is turned to the station mode and connects to a legitimate AP. We use a tool called macchanger to spoof the MAC address of a legitimate AP. A station connected to the rogue AP will be assigned an IP address as if it obtains it from a legitimate AP.

Tunneling the two interfaces of the rogue AP is achieved by adding rules in *iptables*. The adversary's strategies mentioned before are implemented by netfilter/iptables.

(d) Traffic load: We use channel utilization as in [21] to quantify the traffic load. Particularly, the channel utilization during a certain period of time is computed by adding (1) the

time spent by the on-air transmission of all data (including retransmitting), management, and control frames, and (2) the overhead for each frame, such as DIFS and SIFS. This overhead is a part of channel utilization, since the channel remains unavailable in that period. In our testbed, the traffic generator sends constant bit rate (CBR) packets to generate the required *channel utilization* (traffic load).

VI. Evaluation

Here, we present the experimental results of our rouge AP detection scheme. For all our experiments, we use the setup found in Fig 4.

A. Data transmission rate

We begin with investigating whether a rogue AP can manipulate its transmission rate to avoid detection. Most wireless devices adopt rate adaptation algorithms to adjust their transmission rate with respect to changing wireless conditions. However, since there are no specifications with regards to rate adaptation in 802.11 networks, the rogue AP is free to use any 802.11 transmission rate to try to avoid detection. The idea is that a rouge may attempt to always use the highest rate when connected to a legitimate AP so as to reduce the RTT.

TABLE III
Comparison between rate adaptation and fixed 54Mbps in idle traffic and good channel quality conditions

RAP	Average RTT after removing outliers (ms)					
Rate adaptation	$\overline{RTT_{probe}}$	0.661	0.659	0.659	0.66	0.657
	$\overline{RTT_{dns}}$	1.827	1.826	1.795	1.789	1.825
	Δt	1.166	1.167	1.126	1.129	1.168
	θ	1.028	1.023	1.021	1.031	1.027
Fixed 54Mbps	$\overline{RTT_{probe}}$	0.661	0.663	0.656	0.664	0.661
	$\overline{RTT_{dns}}$	1.789	1.826	1.785	1.831	1.78
	Δt	1.128	1.163	1.129	1.167	1.119
	θ	1.021	1.027	1.026	1.037	1.020

To test, we first set up a rouge AP to use the default rate adaptation in idle traffic condition when connecting to AP A, and run our detection algorithm. We then repeat the experiment using the same traffic conditions, except we set the rogue AP to always use the highest possible transmission rate of 54Mbps. In both tests, we set $n = 100$ and use DNS server B. Table III shows the results for the two tests, where $\overline{RTT_{probe}}$ ($\overline{RTT_{dns}}$) is average RTT between probe (DNS) request and the response minus the data transmission time (see lines 7-8 in Algorithm 1), $\Delta t = \overline{RTT_{dns}} - \overline{RTT_{probe}}$, and θ is computed according to Eq.(1). In our algorithm, if $\Delta t > \theta$, the tested AP is identified as rogue; otherwise as legitimate. We observe that (1) even if the rogue AP were set to always send at the highest possible rate, we can still detect the rogue AP, and (2) the performance gain by the rogue AP in using a fixed rate appears to be minor, since rate adaptation can quickly converge to use the best possible rate even if the initial rate is much lower. In fact, in a practical environment, using a fixed rate may result in *worse* performance since more packets will be dropped when traffic conditions fluctuates. This is seen in the larger Δt values in fixed rate experiments. These results are omitted due to page limitations. Lastly, since utilizing a fix rate yields no benefits, we let the rogue AP use rate adaptation for the rest of our experiments.

B. Location of DNS server

A variable that may affect the RTT timings is the location of the DNS server. A DNS server that is more hops away from a station may lead to larger RTT values, or vice versa. Thus, a close DNS server will make rogue AP difficult to be detected, and a far away DNS server will cause legitimate AP to be falsely identified as rogue.

To illustrate the delay introduced by multiple hops, we send 64 byte packets to a local host located at two, four, and five hops away from the station, and measure the time taken for the host to respond. Fig. 6 shows the results. We find that the time delay resulting from additional hops is very small.

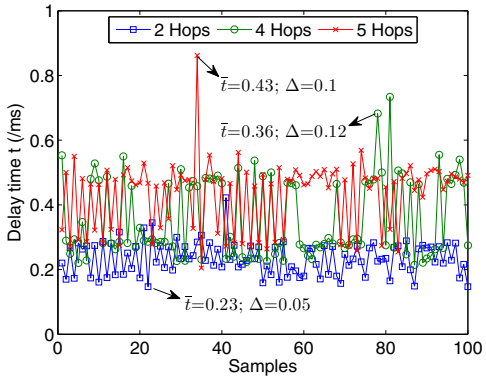

Fig. 6. Delay for transmitting 64 byte packets via different number of hops on wired link. $\bar{t}$ is the mean of delay, and Δ is the variance.

Next, we test our detection algorithm when rogue APs connect to different DNS servers in idle traffic condition. In the first test, we let the legitimate AP and the rogue AP both use DNS server A (see Fig. 4). Packets sent by the station need 3 wired hops to reach the DNS server, and 2 wired hops for the response back. In the second test, we have both legitimate and rogue AP connect to DNS server B which is located in the same subnet. Table IV show the results. We see that our algorithm is able to detect the rogue AP even when the DNS server is located at different places.

C. Wireless Traffic

Here we examine the effects of wireless traffic on our detection algorithm. Since we adopt a timing based approach, variations in network traffic may adversely affect our results. To evaluate, we set the rogue AP to use the most favorable conditions to avoid detection. The rogue AP can best avoid detection when it can connect the station to the real AP as fast as possible. We let the connection between the rogue AP and the AP A (Fig 4) be free of any traffic, thus ensuring the fastest possible transmission between the rogue AP and real AP. This connection is set to channel 11. We then test the rogue AP against AP B, both of which are set to channel 1.

TABLE IV
AVERAGE RTT FOR DNS SERVER UNDER IDLE TRAFFIC SITUATION

	Average RTT after removing outliers (ms)					
Legitimate AP A	$\overline{RTT_{probe}}$	0.633	0.633	0.632	0.634	0.634
	$\overline{RTT_{dns}}$	1.152	1.151	1.148	1.156	1.130
	Δt	0.519	0.518	0.516	0.522	0.025
(Server A)	θ	1.014	1.015	1.016	1.016	1.203
Rogue AP	$\overline{RTT_{probe}}$	0.671	0.663	0.664	0.664	0.661
	$\overline{RTT_{dns}}$	2.151	2.219	2.452	2.371	2.455
	Δt	1.48	1.556	1.788	1.707	1.794
(Server A)	θ	1.057	1.045	1.057	1.062	1.062
Legitimate AP	$\overline{RTT_{probe}}$	0.632	0.633	0.633	0.632	0.632
	$\overline{RTT_{dns}}$	0.691	0.687	0.691	0.687	0.692
	Δt	0.059	0.054	0.058	0.055	0.06
(Server B)	θ	1.004	1.004	1.003	1.003	1.004
Rogue AP	$\overline{RTT_{probe}}$	0.661	0.659	0.659	0.66	0.657
	$\overline{RTT_{dns}}$	1.827	1.826	1.795	1.789	1.825
	Δt	1.166	1.167	1.126	1.129	1.168
(Server B)	θ	1.028	1.023	1.021	1.031	1.027

We use a separate laptop as a traffic generator to control the amount of traffic. We experiment over three traffic conditions, idle traffic, half-saturated traffic, and saturated traffic. In all experiments, we set $n = 100$ and use DNS server A.

Idle traffic: Fig. 7 describes the empirical CDF of RTT for legitimate AP and rogue AP measured in one experiment. The complete results are listed in Table IV. As we can see,

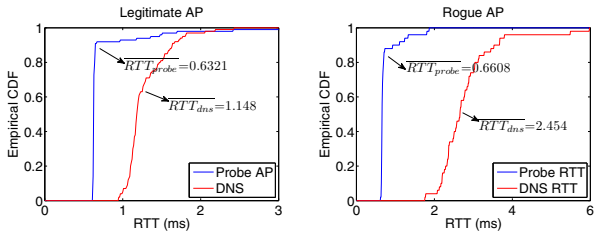

Fig. 7. Empirical CDF of RTT for legitimate AP A and rogue AP in idle traffic situation, while DNS server is A, transmission rate is automatic, and $n = 100$.

the value of Δt is small, and the θ varies a little in idle traffic situation. For the legitimate AP, all Δt are smaller than 1ms, whereas all Δt for the rogue AP are larger than 1ms. Our scheme achieves nearly 100% accuracy, and the total detection time is no longer than 1s.

Half-saturated traffic: We define a half-saturated traffic condition when the ratio of on-air time of all transmitted packets to the total time is 45%. The traffic generator will periodically send packets to AP B to create this condition. The experiment is then repeated to test our algorithm. We find that as the traffic load increases, the average RTT for both probe and DNS messages also increase. At the same time, our algorithm is still able to identify rogue AP with high probability. Fig. 8 illustrates CDF for one experiment. The details are described in Table V.

Saturated traffic: Here, we let the traffic generator send enough packets to create a 90% channel utilization before starting the experiments. Fig. 9 and Table VI describe the results. We find that under heavy traffic conditions, the RTT

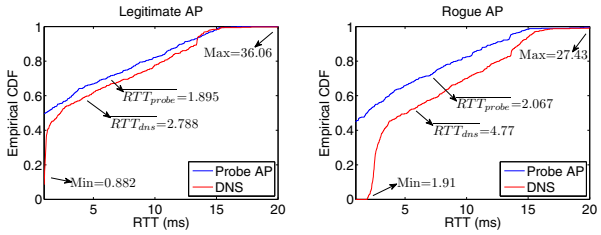

Fig. 8. Empirical CDF of RTT for legitimate AP B and rogue AP in half saturated traffic situation, while DNS server is A, transmission rate is automatic, and $n = 100$.

TABLE V
AVERAGE RTT IN HALF SATURATED TRAFFIC SITUATION

	Average RTT after removing outliers (ms)					
Legitimate AP B	$\overline{RTT_{probe}}$	1.895	1.67	2.009	1.664	1.787
	$\overline{RTT_{dns}}$	2.788	3.486	3.291	3.415	3.133
	Δt	0.893	1.816	1.282	1.751	1.346
(Server A)	θ	1.956	1.972	2.025	1.986	1.985
Rogue AP	$\overline{RTT_{probe}}$	1.744	2.067	2.69	2.749	1.982
	$\overline{RTT_{dns}}$	4.692	4.77	6.215	6.222	4.838
	Δt	2.948	2.703	3.525	3.473	2.856
(Server A)	θ	2.021	2.034	2.296	2.356	2.072

variance for both probe request and DNS lookup are very large. The values range from several milliseconds to hundreds of milliseconds. As a result, some of the legitimate APs may be incorrectly classified as a rogue AP.

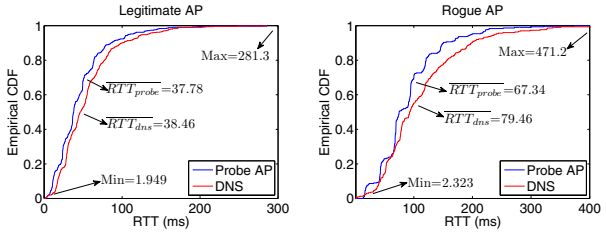

Fig. 9. Empirical CDF of RTT for legitimate AP B and rogue AP in saturated traffic situation, while DNS server is A, transmission rate is automatic, and $n = 100$.

TABLE VI
AVERAGE RTT IN SATURATED TRAFFIC SITUATION

	Average RTT after removing outliers (ms)					
Legitimate AP B	$\overline{RTT_{probe}}$	37.78	41.07	52.55	54.63	**29.36**
	$\overline{RTT_{dns}}$	38.46	45.22	61.28	64.43	**41.72**
	Δt	0.68	4.15	8.73	9.80	**12.36**
(Serv A)	θ	7.70	11.57	14.12	13.49	**9.56**
Rogue AP	$\overline{RTT_{probe}}$	53.04	63.18	67.34	59.18	**54.95**
	$\overline{RTT_{dns}}$	69.13	82.34	79.46	77.59	**69.36**
	Δt	16.09	19.16	12.12	18.41	**14.41**
(Serv A)	θ	12.42	13.29	11.73	12.86	**15.33**

D. Number of samples

Previously, we found that our algorithm does not work well when the wireless traffic is saturated. Here, we explore the effects of testing more times. In other words, we want to determine the effects when we increase from the original

value $n = 100$ to larger n values. We test the legitimate AP and rogue AP separately under different channel utilization. Both tests are repeated 10 times. The detection accuracy is the ratio of the number of the tests in which the AP is correctly identified over a total of 20 times. Fig. 10 illustrates the accuracy against channel utilization.

We find that using $n = 100$ achieves 100 percent of detection accuracy in low traffic situation. However, the accuracy falls to 65% as the channel becomes increasingly saturated. When we set $n = 300$, we are able to obtain 80% as the channel saturation increases.

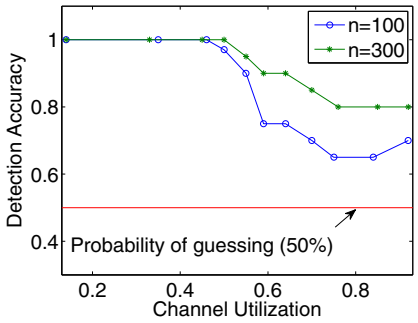

Fig. 10. Detection accuracy against channel utilization

E. Detection time

The detection time is the total amount of time a user has to wait for our algorithm to finish executing. This value is approximately the average RTT multiplied by sample size. When there is heavy traffic, the average RTT increases, and thus the total detection time also increases. Also, we may need to use a larger value for n in heavy traffic which also increases the detection time. Table VII shows the detection time for detecting each AP under different traffic conditions. We omit the time needed to associate with an AP and obtain an IP address since both are necessary regardless of which AP is selected.

TABLE VII
DETECTION TIME UNDER DIFFERENT NETWORK TRAFFIC CONDITION

	Detection time (second)			
	Idle	Half saturated	Saturated	
$n = 100$	0.2	0.9	11.3	AP
	0.3	1.1	17.2	RAP
$n = 300$	0.5	2.7	39.1	AP
	0.8	3.2	50.7	RAP

VII. Conclusion

The ease of setting up a successful rogue AP makes this form of wireless attack a particularly serious security problem. While existing techniques can alleviate this threat, they nonetheless require active participation on the part of the network administrator. In this paper, we present a practical, timing based scheme for the end user to avoid connecting to rogue APs. This is done without any assistance from the network administrator. We implement our approach on commercially available hardware for evaluation.

Acknowledgments

This project was supported in part by US National Science Foundation grants CNS-0721443, CNS-0831904, CAREER Award CNS-0747108, the National High-Tech Research and Development Program of China (863) under Grant No. 2006AA01Z199, the National Natural Science Foundation of China under Grant No. 90718031, No. 60721002, No. 60573106 and the National Basic Research Program of China (973) under Grant No. 2009CB320705.

References

[1] Air defence. [Online]. Available: www.airdefence.net
[2] Air magnet. [Online]. Available: www.airmagnet.com
[3] Air wave. [Online]. Available: www.airwave.com
[4] L. Ma, A. Y. Teymorian, and X. Cheng, "A hybrid rogue access point protection framework for commodity Wi-Fi networks," in *Infocom 2008*.
[5] W. Wei, K. Suh, B. Wang, Y. Gu, J. Kurose, and D. Towsley, "Passive online rogue access point detection using sequential hypothesis testing with TCP ACK-pairs," in *IMC 2007*.
[6] H. Yin, G. Chen, and J. Wang, "Detecting protected layer-3 rogue APs," in *Broadnets 2007*.
[7] S. Shetty, M. Song, and L. Ma, "Rogue access point detection by analyzing network traffic characteristics," in *Milcom 2007*.
[8] P. Bahl, R. Chandra, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill, "Enhancing the security of corporate Wi-Fi networks using DAIR," in *MobiSys 2006*.
[9] A. Adya, P. Bahl, R. Chandra, and L. Qiu, "Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks," in *Mobicom 2004*.
[10] R. Chandra, J. Padhye, A. Wolman, and B. Zill, "A location-based management system for enterprise wireless lans," in *NSDI 2007*.
[11] Y. Sheng, K. Tan, G. Chen, D. Kotz, and A. Campbell., "Detecting 802.11 MAC layer spoofing using received signal strength," in *Infocom 2008*.
[12] S. Jana and S. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," in *Mobicom 2008*.
[13] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Mobicom 2008*.
[14] L. Watkins, R. Beyah, and C. Corbett, "A passive approach to rogue access point detection," in *Globecom 2007*.
[15] 89600s series VXI-based vector signal analyzer. Agilent technologies.
[16] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *Computer Communication Review*, vol. 32, no. 1, p. 74, 2002.
[17] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *Mobicom 2006*.
[18] Madwifi. [Online]. Available: http://madwifi.org
[19] Ipw3945. [Online]. Available: ipw3945.sourceforge.net/
[20] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," in *TOCS 2000*.
[21] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer, "Understanding congestion in IEEE 802.11b wireless networks," in *Internet Measurment Conference*, 2005, pp. 279–292.