A Passive Approach to Wireless Device Fingerprinting

Ke Gao[1], Cherita Corbett[2], and Raheem Beyah[1]
[1]Department of Computer Science, Georgia State University, Atlanta, Georgia 30303
[2]Department of Computer Science, University of California, Davis, California 95616
{kgao, rbeyah}@cs.gsu.edu, corbett@cs.ucdavis.edu

Abstract

We propose a passive blackbox-based technique for determining the type of access point (AP) connected to a network. Essentially, a stimulant (i.e., packet train) that emulates normal data transmission is sent through the access point. Since access points from different vendors are architecturally heterogeneous (e.g., chipset, firmware, driver), each AP will act upon the packet train differently. By applying wavelet analysis to the resultant packet train, a distinct but reproducible pattern is extracted allowing a clear classification of different AP types. This has two important applications: (1) as a system administrator, this technique can be used to determine if a rogue access point has connected to the network; and (2) as an attacker, fingerprinting the access point is necessary to launch driver/firmware specific attacks. Extensive experiments were conducted (over 60GB of data was collected) to differentiate 6 APs. We show that this technique can classify APs with a high accuracy (in some cases, we can classify successfully 100% of the time) with as little as 100000 packets. Further, we illustrate that this technique is independent of the stimulant traffic type (e.g., TCP or UDP). Finally, we show that the AP profile is stable across multiple models of the same AP.

1. Introduction

Fingerprinting networked devices has been around for many years. The general idea is to extract leaked information about the device's software, operating system, or hardware components. This can be accomplished in an *active* or *passive* manner. Active approaches usually entail interrogating a node with various types of packets. These packets may vary in size and can be either legitimate or malformed. The goal of active techniques is to trigger a response that is unique to the entity that is being fingerprinted. Passive techniques are often more desirable to the fingerprinter, however they usually give less information about a node than their active counterpart. Generally, passive approaches do not inject any stimulant into the system of interest, rather they capture data silently with the goal of not alerting or disturbing the system being surveilled. The data is later analyzed to reveal patterns that are unique to the system of interest.

As with any tool, the intentions of the user determine whether the tool is considered bad or good. Accordingly, fingerprinting tools can be used for *offensive* and *defensive* purposes. In this discussion, we assume that offensive use of fingerprinting tools is usually done by the attackers while the good guys usually conduct defensive fingerprinting. Offensive fingerprinting is often done by attackers to gain enough information about a node's characteristics to execute a precise attack on the unsuspecting node. In general, the more the attacker knows about the system that he intends to attack, the more likely he will be successful with the attack. Accordingly, reconnaissance is one of the most time consuming, and most important stages of an attack life cycle. Fingerprinting for defense can be used by network administrators to search for nodes on the network that do not have a particular architecture or configuration. For example, a network administrator may use such a method to search for rogue nodes who gained access to the network using valid credentials (possibly acquired via a phishing attack) and therefore remained undetected by an intrusion detection system.

There have been many vulnerabilities discovered in wireless devices that have sparked increased interest in fingerprinting wireless nodes. The authors of [12] present a kernel-level exploit for 802.11 device drivers on a Windows platform. There are also repositories (e.g., [1], [2], and [8]) of known vulnerabilities for various models of wireless devices that describe, for example, methods to crash, bypass authentication, or completely take control of a device. Table 1 highlights a sample of known vulnerabilities for various access points. The effectiveness of an exploit on these vulnerabilities is hinged upon identifying the type of wireless device so the best attack vector is chosen against the target. On the other hand, it behooves a network administrator to fingerprint and identify rogue devices that are not managed to prevent these targeted attacks.

Table 1. Known Vulnerabilities in APs

Model Number	Description	Impact
Netgear WN802T	Specially crafted EAPoL-Key or association request packets can cause reboot or hang-up	DoS
3Com 8760	Authentication mechanism improperly restricts access to administration pages	Unauthorized access
Cisco 4400	A memory leak in the handling of SSH management connections can cause a crash	DoS
Cisco 1300	Injecting malicious packets can hang up the AP	DoS
Cisco 1130	IOS HTTP server injected with malicious code can invalidate authorization	Unauthorized Access
ASUS WL-500W	Buffer overflow can invalidate authorization	Unauthorized Access
D-Link DWL-1000AP	The administrative password is not stored properly	Unauthorized Access
D-Link DIR-400	Buffer overflow can invalidate authorization	Unauthorized Access
Netgear WNDAP330	Improperly handling user request can hang up AP	DoS
Nortel 2200	Crafted user association request can cause hang-up	DoS
Belkin F5D6130	Crafted SNMP request can cause hang-up	DoS
Proxim AP-4000	Hard coded static WEP key invalidates authorization	Unauthorized Access

In this paper we introduce a blackbox-based wireless device fingerprinting technique that can be used for offensive or defensive purposes. Blackbox testing is a popular technique used to test software where the contents are unknown to the tester [10] (hence the name blackbox). To conduct the test, a stimulant is applied to the input of the software and the output is observed. From this, the tester can infer how the software (i.e., blackbox) acted on the input. In our case, the blackbox is an AP and just as the software testers are not privy to source code during software blackbox testing, we are not privy to the proprietary architecture of the AP. The input to our blackbox is a packet train and the output is the same packet train, however individual elements (i.e., packets) have been shifted in time. The shifting is a result of the internal architecture of the AP. Further, since each AP has a different architecture, this shifting is unique to the AP. We use wavelet analysis to amplify and extract the unique patterns generated by the internals of the AP.

The rest of this paper is organized as follows. In Section 2, we present our contribution in the context of other related works. In Section 3 we present our technique and detection methodology. Section 4 introduces how to generate master signatures and their optimal bin size. In Section 5, we give our experimental setup while Section 6 presents the experimental results. In Section 7, we present current limitations of our technique. The paper is concluded and future work is given in Section 8.

2 Related Work

The existing work in this area can be placed into three categories: OS fingerprinting, host fingerprinting, and device type / driver fingerprinting.

The first category, OS fingerprinting, deals with looking at different features of the protocol stack to differentiate different operating systems. Nmap [4] and Xprobe [9] are active fingerprinting tools that create special testing packets to determine which OS is being used by the target. In contrast to the aforementioned active approaches, p0f [6] uses TCP/IP protocol information to passively determine the OS. SinFP [7] is a new approach to OS fingerprinting that bypasses some of the emerging limitations of Nmap (i.e., working with PAT/NAT configurations and emerging packet normalization technologies). SinFP has both an active and a passive mode. Although these methods are effective for OS fingerprinting, the goal of our work is to differentiate wireless hardware devices.

Another body of work that is relevant to the proposed work deals with fingerprinting specific hosts. Seminal work in this area was introduced by Kohno et al. in [21]. In [21], a method for remotely fingerprinting a physical device by exploiting the implementation of the TCP protocol stack was proposed. The authors use the TCP timestamp option of outgoing TCP packets to reveal information about the sender's internal clock. The authors' technique exploits microscopic deviations in the clock skews to derive a clock cycle pattern as the identity for a device. The authors of [20] take a similar approach of that in [21] (i.e., using clock skew to uniquely identify nodes), however the goal of [21] is to uniquely fingerprint APs. Also, instead of getting the timestamp from TCP packets, they obtain the timestamp from 802.11 beacon frames. The aforementioned works seek to uniquely identify specific devices. In contrast, our work focuses on identifying specific device types. Identifying different types of nodes rather than specific nodes enables a profile to be built for a group of nodes rather than for a single node. This profile can then be used to predict the behavior of an entire cohort.

Another category of work focuses on fingerprinting device type / driver. In [13, 14, 15, 16], we discuss passive techniques for identifying the type of wireless card used by a node during normal node operation. Spectral analysis is used to determine identifying frequencies associated with ambiguously specified medium access control (MAC) functions in the 802.11 standard. Similar to the work done in [13, 14, 15, 16] the authors of [18] use statistical information of the 802.11 management frame transmission to fingerprint the device type / driver. This work focuses on sta-

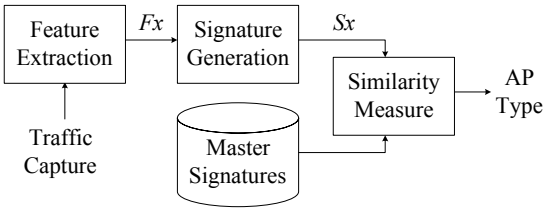


Figure 1. Overview of our approach

tistical patterns generated as a result of different software components of network cards. In contrast, our work seeks to determine how the hardware of an AP uniquely modulates an input signal in order to fingerprint the AP. Further, the techniques proposed in [13, 14, 15, 16, 18] are limited to NICs, where our technique can be extended to any architecturally different wireless device. The work that is the most closely related to our work is that proposed in [11]. In [11], the authors fingerprint wireless APs by probing them with various regular and malformed packets. However, this technique is active and can be easily detected by a malicious user who controls the access point.

In this paper we present a passive wireless device type fingerprinting approach that uses a blackbox-based paradigm. This technique can be used for offensive or defensive purposes and is extensible to any wireless device.

3 Detecting Methodology

3.1 Problem Definition

Given a set of APs for which master signatures are known, we seek to identify an unknown AP based on the observation of its egress traffic. This problem is more formally stated as follows. We have a candidate set of APs, A, where $A = \{a_1, a_2, ..., a_n\}$ and n is the number of unique types of APs. For each type of AP, a_i, where $a_i \epsilon A$, there is a feature tuple $F_i = \{T_i, b_i\}$ that distinguishes it from all other APs in the set A. T_i is the sequence of packet inter-arrival time (IAT) values used as the input signal for signature generation and b_i is the corresponding optimal bin size parameter for sampling the signal. Using the feature tuple $F_i = \{T_i, b_i\}$ we derive a master signature s_i for each a_i, resulting in a set of master signatures $S = \{s_1, s_2, ..., s_n\}$.

For traffic captured from an unknown AP, a_x, we extract a sequence of IAT values to form a signal denoted as T_x. Using its feature T_x, we seek to identify a_x. Similarly, we need to generate a signature, denoted as s_x, for the unknown AP type, a_x. Then we perform a pairwise comparison of the unknown signature s_x with all the master signatures $\forall s_i \epsilon S$. For the s_i that is the *closest* to s_x, the corresponding type a_i is concluded to be type for a_x. An overview of our approach is demonstrated in Figure 1. Details of our proposed solution are provided in the following sections.

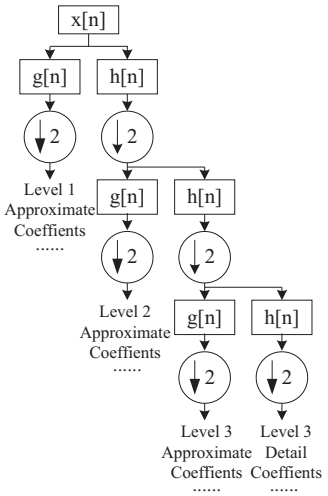


Figure 2. Wavelet filter

3.2 Feature Extraction

Before feature extraction can occur, an output from the unknown AP that we seek to identify must be observed. To do so, we send traffic that emulates normal data transmission through the AP to a destination controlled by the fingerprinter for traffic capturing. Note, we do not use specially crafted packets to probe or change the operation of the AP. Thus, we remain transparent while obtaining data for fingerprinting.

Once traffic is captured, the fingerprinter needs to extract features that will help to identify the type of AP. The feature chosen to characterize the network traffic behavior is the packet IAT. IAT measures the delay (Δt) between successive packets, thus characterizing the rate of the traffic flow. Using packet IATs, we produce a signal $T = (\Delta t_1, \Delta t_2, ..., \Delta t_p)$ where p is the number of packets captured. We assert that the traffic rate characterized by the IATs is influenced in a systematic manner by the architecture of the AP, which differs depending on its type. The results in Section 6 illustrate the viability of this assertion.

3.3 Signature Generation

We use wavelet analysis in our approach for signature generation due to its well-known capability for multi-resolution decomposition suited for analysis of non-stationary signals. For additional understanding of wavelets refer to [22]. Wavelet analysis has proven to reveal discriminating attributes of a system that are not directly observable. Wavelet analysis of network traffic [17, 19] has relied on

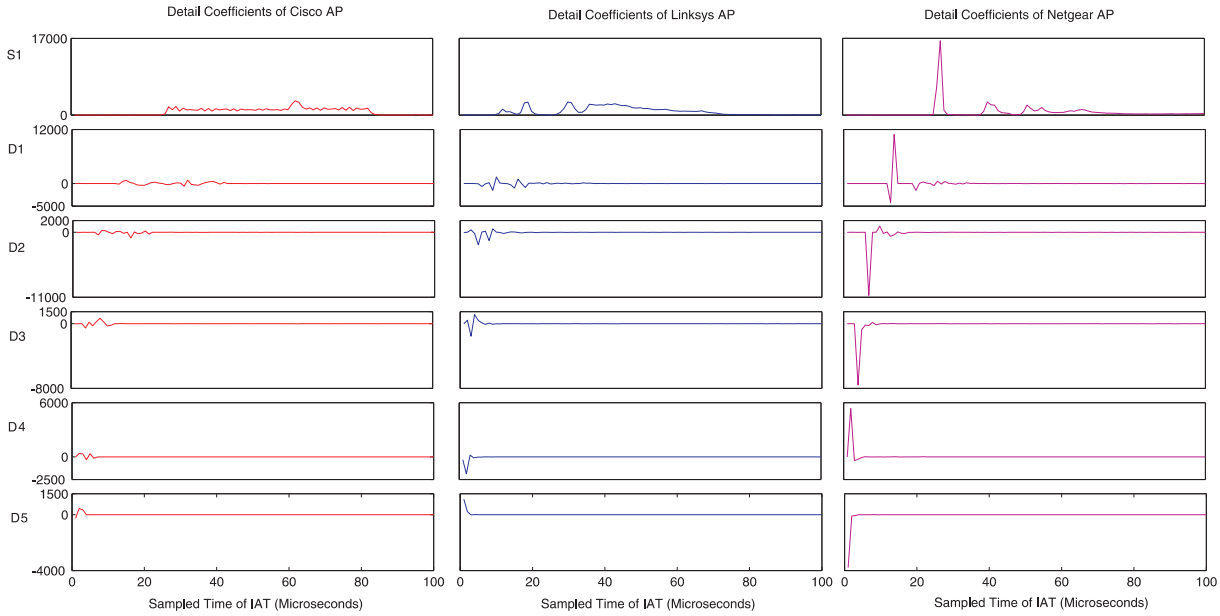


Figure 3. Detailed coefficients of a signal from a Cisco AP, Linksys AP and Netgear AP

this property to detect attacks, characterize wireless performance, etc. Here we apply the wavelet transform to the IAT signals, decomposing them into wavelet coefficients. The corresponding wavelet coefficients constitute the signature and are used for comparison to identify an AP.

Our implementation of the wavelet transform is based on a filter bank of high-pass ($h[n]$) and low-pass ($g[n]$) filters as illustrated in Figure 2. At each level, the low-pass filter produces approximate coefficients and the high-pass filter produces detail coefficients. The outputs are downsampled by 2. The high-pass results recursively become the input into the next level. To construct the filter bank we use the Haar wavelet. The approximate and detail coefficients of the Haar wavelet at level j are defined by equations 1 and 2 respectively. The subscript k refers to the index of the coefficient vector.

$$a_{j,k} \equiv \frac{1}{\sqrt{2}}(X_{2k-1} + X_{2k}) \tag{1}$$

$$d_{j,k} \equiv \frac{1}{\sqrt{2}}(X_{2k-1} - X_{2k}) \tag{2}$$

For our analysis, based on extensive experiments, we retain the detail coefficients and discard the approximate coefficients. As an example, Figure 3 shows the detail coefficients of the IAT signals from Cisco, Linksys, and Netgear APs. The signature for AP a_i is denoted as $s_i = (d_1^i, d_2^i, ...d_j^i)$, where d_j^i is the vector of detail coefficients produced at level j. We use the same approach to generate the master signatures for our candidate group of APs as we do for the unknown AP that we wish to identify. However, the signatures for the candidate group are pre-computed from a set of traffic captures used to train our system and are not a part of the experimental analysis. The master signatures are stored in a repository for matching against unknown APs.

To help determine the best wavelet levels to use for signatures, we evaluated the energy concentrated at the different levels for an arbitrary set of traffic traces generated from candidate APs. The wavelet energy is defined as the sum of the square of detailed wavelet coefficients divided by the length of the vector. This can be expressed mathematically as:

$$E_j = \frac{1}{N_j} \sum_k |d_{j,k}|^2 \tag{3}$$

Figure 4 shows the wavelet energy for all six APs. It can be observed that the difference in the energy among the APs are the most pronounced at wavelet levels 2, 3, 4, and 5. Based on this evaluation, we selected the detail coefficients from levels 2, 3, 4, and 5 to generate the signature. Therefore, the signature becomes $s_i = (d_2^i, d_3^i, d_4^i, d_5^i)$.

3.4 Similarity Measure

Using the signatures derived from the wavelet detail coefficients, we measure the similarity of the unknown to the candidate set of APs. Cross-correlation is commonly used to calculate the similarity between two signals for applications in pattern matching. For our analysis we implement

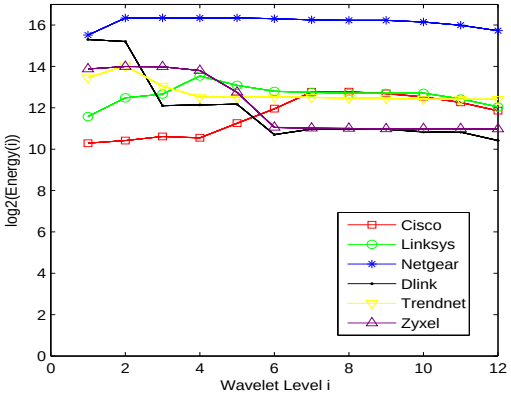


Figure 4. Energy graph of six APs

the circular cross-correlation (cxcorr) shown in algorithm 1. For two vectors of detail coefficients d^x and d^i, left-shift d^i in a circular manner, then perform the dot product of d^x with the shifted d^i. The result is a sequence of l values, where l is the length of d^i. The maximum value of this sequence is assigned to the correlation parameter, c. That is,

$$c_j^i = max\{cxcorr(d_j^x, d_j^i)\}, 0 \leq c_j^i \leq 1 \quad (4)$$

where c_j^i is the measure of similarity of the detail coefficients of the unknown a_x to a candidate a_i, where $a_i \epsilon A$, at level j. It follows that the total similarity measure, sim_i, is the sum of the correlation measure of the unknown a_x to the candidate a_i across all j levels:

$$sim_i = \sum_{j=2}^{5} c_j^i \quad (5)$$

For each AP in which we have a master signature, we use algorithm 2 to calculate the total similarity between a_x and a_i, $\forall a_i \epsilon A$. The a_i that has the greatest sim_i is assigned as the type for a_x. Table 2 shows the correlation c_j^i per level between different APs. For APs that are of the same type, the similarity approaches 1 across all levels, whereas the correlation is lower at levels 2 through 5 for APs that are of a different types. This observation further supports the decision to use levels 2-5 for distinguishing APs.

Algorithm 1 maxCXCORR(d_x, d_i)

1: **for** $l = 1 : length(d_i)$ **do**
2: $cxcorr(l) = d_x * d_i'$
3: $d_i = [d_i(end)d_i(1 : end - 1)]$ % circular shift
4: **end for**
5: $c = max(cxcorr)$
6: return c

Algorithm 2 sim(T_x, T_i)

1: $T_x = sampleData(T_x, binsize)$
2: $T_i = sampleData(T_i, binsize)$
3: $sim = 0$
4: $[g_x, h_x] = haar(T_x)$
5: $[g_i, h_i] = haar(T_i)$
6: **for** $j = 2 : 5$ **do**
7: $[g_x, h_x] = haar(h_x)$
8: $[g_i, h_i] = haar(h_i)$
9: $sim = sim + maxCXCORR(h_x, h_i)$
10: **end for**
11: return sim

3.5 Summary of Approach

Our methodology uses the following steps:

Step 1: For a given traffic capture from an unknown AP type a_x, extract sequence of IAT values to generate the feature signal T_x.

Step 2: Apply wavelet transform to decompose the signal T_x into detail coefficients at different levels j. Use the coefficients to form the signature $s_x = (d_1^x, d_2^x, ..., d_j^x,)$.

Step 3: Measure the similarity of s_x to the set of master signatures S using circular cross-correlation.

Step 4: Select the AP type a_i that has the highest similarity measure and assert a_i to be the identity of a_x.

Table 2. Similarity measure between APs

Level	maxCXCORR Dlink1 vs Dlink2	maxCXCORR Dlink1 vs Cisco	maxCXCORR Dlink1 vs Netgear
1	0.991	0.99	0.97
2	0.995	0.611	0.627
3	0.993	0.781	0.751
4	0.996	0.812	0.821
5	0.991	0.824	0.822
6	0.992	0.991	0.993

4 Master Signatures

Now that we have described our approach, let us take a moment to describe the process used to derive the master signatures for the candidate set of APs listed in Table 3. As stated earlier, the master signatures are used for similarity testing to identify an unknown AP.

For each AP, we generated 10 packet traces each containing 100,000 packets. From each trace, we extracted the IAT values as described in Section 3.2 to generate the feature signal T. The process of deriving the master signatures is two-fold. First we must determine the bin size b_i used to sample the feature signal T for each particular type of AP.

Once we determine the best bin size, we select 1 out of the 10 packet traces to use for generating the master signature.

We empirically evaluated 10 different bin sizes from 1 us to 10 us, in 1 us intervals. For each bin size we perform a pairwise comparison between packet traces from one AP to traces from the other remaining APs, maintaining a cumulative sum of the similarity measure for each pair across the set of 10 traces. The bin size that resulted in the lowest cumulative similarity is chosen as the best bin size. That is, the optimal bin size is the bin size that generated signatures that were the most distinct from the other signatures. Algorithm 3 demonstrates this process, where the input variable T is the set of feature signals from all the APs.

Using the best bin size, determined by Algorithm 3, we seek to find the individual packet trace from the set of 10 that is best for generating the master signature to represent each AP. For each packet trace, we measure its similarity to the traces from the remaining APs. The packet trace out of the set of 10 that causes the least similarity is chosen to generate the master signature. This process is shown in Algorithm 4.

Table 3. AP list

Brand	Model Number
Cisco	Aironet 1130 AG
Netgear	WAG102
Linksys	WAP54G
D-Link	DWL-2100AP
TRENDnet	TEW 434APB
ZyXEL	G-570S

Algorithm 3 Find_Best_Binsize(T)

$numAP = 6$;
$numTraces = 10$
$binsize = (1:10) * 10^{-6}$
$s = zeros(size(binsize))$
for $m = 1 : numAP$ **do**
  $n = [(1 : m-1)\ (m+1 : numAP)]$
  for $b = 1 : length(binsize)$ **do**
    for $t_m = 1 : numTraces$ **do**
      for $j = 1 : (numAP - 1)$ **do**
        for $t_n = 1 : numTraces$ **do**
          $s(b) = s(b) + sim(T(m, t_m), T(n(j), t_n), binsize(b))$
        end for
      end for
    end for
  end for
  $[value, index] = min(s)$
  $best\_binsize(m) = binsize(index)$
end for
return $best\_binsize$

Algorithm 4 Build_Master_Signature(T)

$numAP = 6$
$numTraces = 10$
$s = zeros(size(numTraces))$
$bestBinsize = Find\_Best\_Binsize(T)$
for $m = 1 : numAP$ **do**
  $n = [(1 : m-1)\ (m+1 : numAP)]$
  for $t_m = 1 : numTraces$ **do**
    for $j = 1 : (numAP - 1)$ **do**
      for $t_n = 1 : numTraces$ **do**
        $s(t_m) = s(t_m) + sim(T(m, t_m), T(n(j), t_n), bestBinsize(m))$
      end for
    end for
  end for
  $[value, index] = min(s)$
  $masterTrace(m) = T(m, index)$
end for
return $masterTrace$

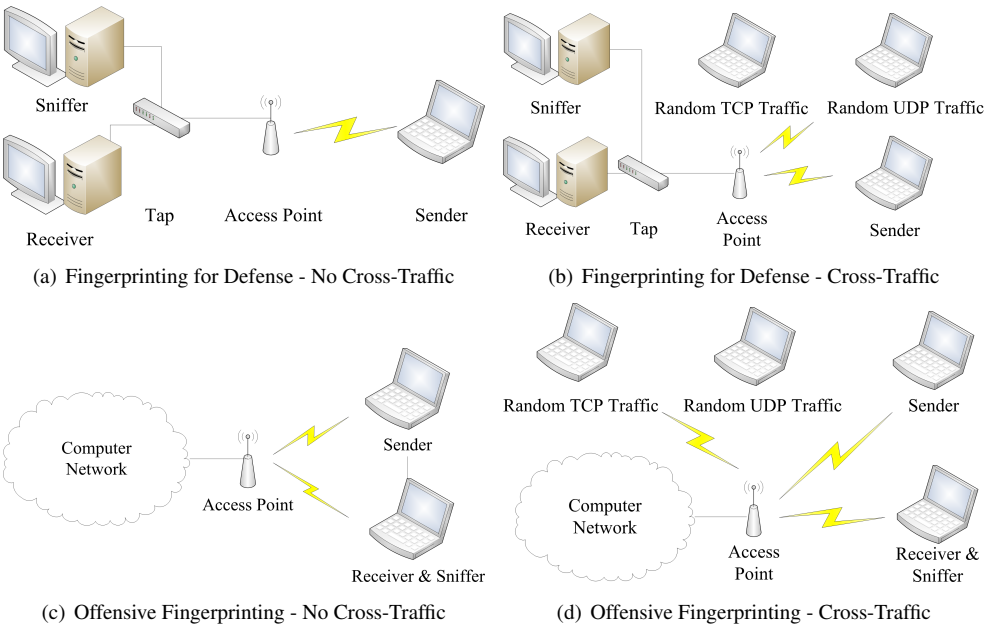
5 Experimental Setup

In this section we discuss the experimental setup. Separate scenarios were generated for defensive and offensive fingerprinting with our technique.

5.1 Scenario 1 - Fingerprinting for Defense

5.1.1 Defense Model

From a defensive perspective, the objective is to identify wireless devices (independent of the end user) that do not belong in an effort to preserve the security of the network. A wireless device unknown to a network administrator, malicious or benign, can expose a network to vulnerabilities and attacks, thus compromising the security of the entire network and its users. We put forth a defense model that presents the assumption of the capabilities of the network



(a) Fingerprinting for Defense - No Cross-Traffic

(b) Fingerprinting for Defense - Cross-Traffic

(c) Offensive Fingerprinting - No Cross-Traffic

(d) Offensive Fingerprinting - Cross-Traffic

Figure 5. Experiment Environment

administrator. In this scenario, it is expected that there will be traffic traversing the unknown wireless device to the wired segment of the network without the need to manufacture stimulant traffic by the fingerprinter. We assume that the network administrator can passively listen to traffic traversing the wired side (originating from the wireless side) of the network using a mirror port on a switch. Based on the egress traffic observed from an unknown wireless device, the network administrator seeks to determine its identity. Establishing this identity enables an administrator to control access as well as execute countermeasures (e.g., to take device offline).

5.1.2 Experimental Setup

To emulate normal traffic, the sender (wireless side) and receiver (wired side) run iperf [3]. The sender sends data packets to the receiver on the designated port. The data is passively monitored using a physical tap (nTAPs [5]). However, in an actual deployment it is expected that the system administrator will capture data on a mirror port of the switch. We set the transmission rate to 1Mbps for both the UDP and TCP experiments. The sender sends data packets to the receiver on the designated port. We set the TCP maximum segment size (MSS) to 100 for the TCP experiments and the packet size is set to 50 bytes in the UDP experiments. These small packets cause more packets to be sent to meet the specified data rate of 1Mbps. This inundation of packets put stress on the APs and allow differentiating aspects to present themselves in the resultant packet train. The traffic is passively monitored on the wired side using a sniffer.

To mimic real traffic and real scenarios we generate TCP and UDP traffic. Additionally, we generate traffic flows without encryption, with wired equivalency protocol (WEP) encryption, and wifi protected access (WPA). Finally, we consider the aforementioned combinations of traffic types without (Figure 5(c)) and with (Figure 5(d)) competing wireless traffic (TCP and UDP traffic with random rate distributions). For the scenarios that included cross-traffic, we follow the same method as above, however we add random traffic that goes through the AP. This scenario is closer to a real scenario as it contains traffic other than that used for fingerprinting. One laptop is used to generate TCP traffic. Another is for generating UDP traffic. Both laptops generate traffic with a random transmission rate ranging from 0 to 200Kbps that bursts repeatedly for 5 seconds. Table 4 summarizes the different combinations of traffic types used.

5.2 Scenario 2 - Offensive Fingerprinting

5.2.1 Threat Model

From an offensive perspective, the objective is to gather attack-relevant information on a target wireless device through network-based surveillance. An attacker seeks to increase its ability to gain a foothold and penetrate a defended network. We assume the attacker is able to associate with the wireless device (i.e., AP) using legitimate credentials. The attacker could use stolen (e.g., via phishing) cre-

Table 4. Summary of different combinations of traffic types

Case #	Transport Protocol	Encryption	Competing Traffic
1	UDP	None	No
2	UDP	WEP	No
3	UDP	WPA	No
4	TCP	None	No
5	TCP	WEP	No
6	TCP	WPA	No
7	UDP	None	Yes
8	UDP	WEP	Yes
9	UDP	WPA	Yes
10	TCP	None	Yes
11	TCP	WEP	Yes
12	TCP	WPA	Yes

dentials from an authorized user or have its own user access (e.g., AT&T account to access the network at Starbucks). We further assume that the attacker has two network interface cards (using two cards in one machine or two separate machines) that can be used to independently send and receive traffic. The attacker also has standard capabilities of capturing traffic using a wireless sniffer. The attacker does not have access to the wired side of the network so fingerprinting must be done using the wireless segment only. To do so, the attacker sends wireless traffic from one NIC through the target wireless device back to itself at the second NIC. Unlike scenario 1, the attacker has to generate artificial (legitimate) traffic in order to have observable traffic from the target. However, the traffic is transparent to the target because the attacker does not use specially crafted packets to probe or change the operation of the target AP.

5.2.2 Experimental Setup

To emulate normal traffic, the sender (wireless side) and receiver (wireless side) run iperf [3]. The sender sends data to the receiver on the designated port. The data is passively monitored wirelessly by the attacker's card that is designated for receiving. As in scenario 1, to emulate real traffic we generate TCP and UDP traffic. We set the transmission rate to 1Mbps for both the UDP and TCP experiments. The sender sends data packets to the receiver on the designated port. We set the TCP maximum segment size (MSS) to 100 for the TCP experiments and the packet size is set to 50 bytes in the UDP experiments. These small packets cause more packets to be sent to meet the specified data rate of 1Mbps. This inundation of packets put stress on the APs and allow differentiating aspects to present themselves in the resultant packet train. Additionally, we generate traffic flows without encryption, with wired equivalency protocol (WEP) encryption, and wifi protected access (WPA). Finally, we consider the aforementioned combinations of traffic types without (Figure 5(c)) and with (Figure 5(d)) competing wireless traffic (TCP and UDP traffic with random rate distributions). For the scenarios that included cross-traffic, we follow the same method as above, however we add random traffic that traverses the AP. This scenario is closer to a real scenario as it contains traffic other than that used for fingerprinting. One laptop is used to generate TCP traffic. Another is for generating UDP traffic. Both laptops generate traffic with a random transmission rate ranging from 0 to 200Kbps that bursts repeatedly for 5 seconds. Table 4 summarizes the different combinations of traffic types used.

5.3 Data Collection

In our experiments, each trace contains 100,000 packets and is approximately 1 minute long. We conduct experiments in twelve cases for scenario one (traffic between a wireless node and a wired node) as well as twelve cases for scenario two (traffic between two wireless nodes or cards) as listed in Table 4. Thus a total of 24 different cases were considered.

As mentioned in Section 4, for each case a master signature (24 master signatures) was created by training our classifier using 10 traces/case. Therefore, there are 6APs, 24 cases, 10 traces/case so a total of 1440 traces are used for training. Next, we randomly choose another 100 unknown traces for each case that is composed of traces created by different APs. So in scenario one, there are a total of 120 traces per access point (10 for each of the 12 cases in scenario 1) to develop the 12 master signatures and 1200 traces (about 30 gigabytes of data) to test the accuracy of the classifier. Scenario 2 also has 120 traces per access point to develop the corresponding 12 master signatures and the remaining 1200 traces (another 30 gigabytes of data) to test the accuracy of the classifier.

6 Experiment Results

In this section we present results that show the effectiveness of our technique given the aforementioned scenarios.

6.1 Overall Results

Figures 6, 7, 8, and 9 illustrate the overall effectiveness of our technique. Specifically, the figures show that the accuracy of every scenario approaches 100% as the amount of traffic used to fingerprint the AP increases. Additionally, the results show that our technique can successfully

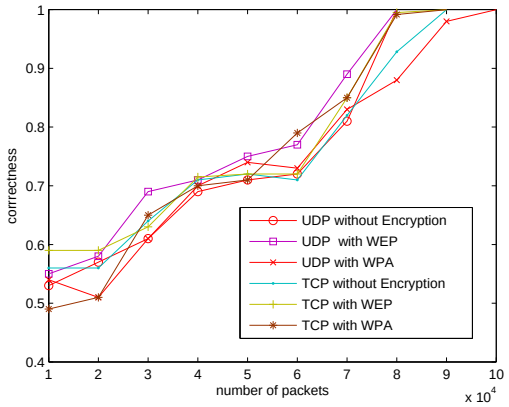


Figure 6. Fingerprinting for Defense - No Cross-Traffic

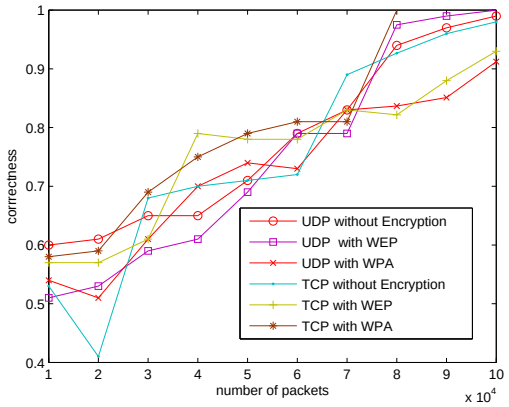


Figure 7. Fingerprinting for Defense - Cross-Traffic

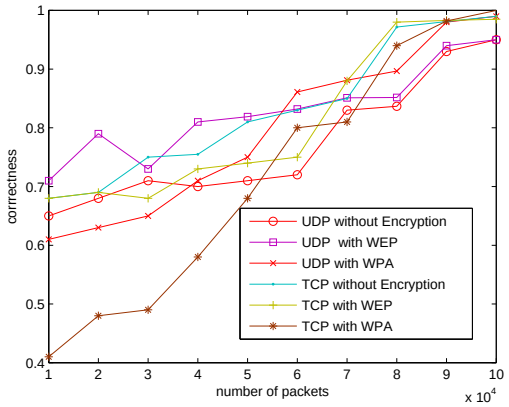


Figure 8. Offensive Fingerprinting - No Cross-Traffic

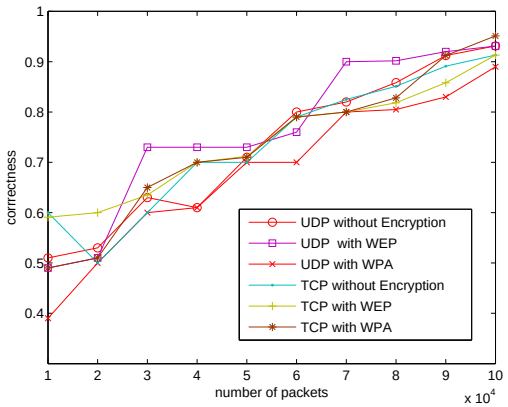


Figure 9. Offensive Fingerprinting - Cross-Traffic

fingerprint APs independent of encryption, transport protocol, and background traffic on the wireless network. As expected, the fingerprinting for defense scenarios (Figures 6 & 7) are more accurate than the offensive fingerprinting scenarios (Figures 8 & 9). This is because in the defensive scenario (scenario 1) the network administrator is capturing packets on the wired link, thus the wireless traffic traverses the wireless link once. However, in the offensive scenario (scenario 2) the attacker captures the resultant packet train on the wireless side. Thus, in addition to the input and output signals interfering with each other (because the attacker is sending and receiving at the same time), there is an additional variable delay (resulting from its medium access control protocol) from accessing the wireless link twice.

6.2 Stability Across Multiple Instances of the Same Access Point

To ensure that our technique can identify general types of APs (not just a specific device), we conducted experiments with several types of APs for which we had two of each. The APs used in these experiment are Dlink, Trendnet, and Zytel. Figure 10 gives the corresponding energy graphs for each pair of APs. The figure shows that each instance of an AP type has nearly the same energy graph, indicating that each instance of the AP type acted similarly upon the packet train. Thus, our classifier would classify exact models of AP the same. This is important if one seeks to fingerprint a cohort of devices as opposed to one specific device.

7 Limitations

As with every scheme, there are limitations with ours. Although, the results were promising the set of APs used

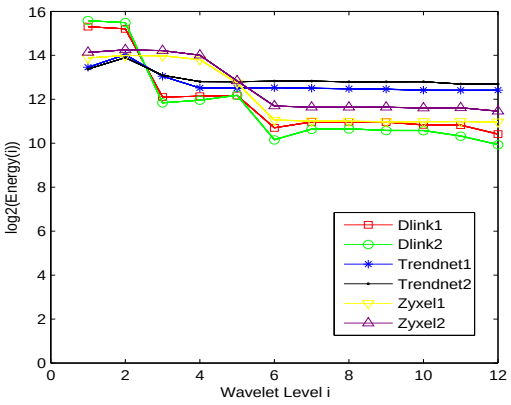


Figure 10. Energy graph of APs of three types)

was limited. Additionally, the experiments were conducted using a wireless testbed with emulated traffic as opposed to a live network with real traffic.

8. Conclusion and Future Work

In this paper we present a passive blackbox-based approach to AP fingerprinting that can be used for offensive or defensive purposes. To our knowledge this is the first approach of its kind. We illustrate, with extensive experiments (over 60GB of data), the efficacy of the proposed technique in a multitude of scenarios. In the future we plan to extend this technique to fingerprinting other wireless devices (e.g., smartphones). The aforementioned limitations will also be addressed in our future work.

9. Acknowledgment

This work was partly supported by NSF Grant No. CAREER-CNS-844144.

References

[1] Access point vulnerabilities, 2009. http://www.f-secure.com.

[2] Access point vulnerabilities, 2009. http://www.securityfocus.com.

[3] Iperf: Network testing tool, 2009. http://sourceforge.net/projects/iperf/.

[4] Nmap: Free security scanner for network exploration & security audits, 2009. http://www.nmap.org/.

[5] ntap: Network data capture tool, 2009. http://www.networkinstruments.com/products/ntaps/10_100.html.

[6] p0f: A versatile passive os fingerprinting tool, 2009. http://www.gomor.org/bin/view/Sinfp/WebHome.

[7] Sinfp: A new approach to os fingerprinting, 2009. http://lcamtuf.coredump.cx/p0f.shtml.

[8] Vulnerabilities of netgear wndap330, 2009. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0052.

[9] Xprobe: Active os fingerprinting tool, 2009. http://xprobe.sourceforge.net/.

[10] B. Beizer. *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[11] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles. Active behavioral fingerprinting of wireless devices. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pages 56–61, New York, NY, USA, 2008. ACM.

[12] J. Cache, H. D. Moore, and Skape. Exploiting 802.11 wireless driver vulnerabilities on windows, January 2009. http://www.uninformed.org/.

[13] C. L. Corbett, R. A. Beyah, and J. A. Copeland. A passive approach to wireless nic identification. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 5, pages 2329–2334, June 2006.

[14] C. L. Corbett, R. A. Beyah, and J. A. Copeland. Using active scanning to identify wireless nics. In *Information Assurance Workshop, 2006 IEEE*, pages 239–246, June 2006.

[15] C. L. Corbett, R. A. Beyah, and J. A. Copeland. Passive classification of wireless nics during active scanning. *International Journal of Information Security*, 2008:335–348, 2008.

[16] C. L. Corbett, R. A. Beyah, and J. A. Copeland. Passive classification of wireless nics during rate switching. *EURASIP J. Wirel. Commun. Netw.*, 2008:1–12, 2008.

[17] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger. Dynamics of ip traffic: a study of the role of variability and the impact of control. *SIGCOMM Comput. Commun. Rev.*, 29(4):301–313, 1999.

[18] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, Berkeley, CA, USA, 2006. USENIX Association.

[19] P. Huang, A. Feldmann, W. Willinger, and P. H. A. Feldmann. A non-intrusive, wavelet-based approach to detecting network performance problems, 2001.

[20] S. Jana and S. K. Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 104–115, New York, NY, USA, 2008. ACM.

[21] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 211–225, Washington, DC, USA, 2005. IEEE Computer Society.

[22] M. Weeks. *DIGITAL SIGNAL PROCESSING Using MATLAB and Wavelets*. INFINITY, 2007.