Boomerang: Acoustic Fingerprinting for Network Access Control

Peter Hornyack Nicholas Hunt

Computer Science and Engineering, University of Washington

1. Introduction

As the prevalence of wireless networking has increased, so has the need for easy-to-use methods of securing wireless networks from abuse by unauthorized users. This paper introduces the distinction between the *wireless coverage area* of an access point, and the *wireless service area*, over which a network operator wishes to provide wireless service. This paper also presents Boomerang, a system that allows network owners to define the wireless service area through acoustic fingerprinting techniques. Boomerang's design is unique in that it restricts wireless network access to clients that are within acoustic range of the access point, which is a desirable feature in settings such as coffee shops and apartments. Further, Boomerang can be used in conjunction with existing access control methods such as WPA and WEP to provide further access restriction and confidential communication between clients and the access point.

1.1 Motivation

Owners of wireless access points (WAPs) require access control mechanisms to prevent unauthorized users from utilizing the network. Unauthorized users not only consume network bandwidth and reduce performance, but may also present security risks to legitimate users. An example of when such access control mechanisms are especially valuable is in a coffee shop, where wireless Internet access is often provided as a service for customers. Complicating matters, however, is the fact that the range of wireless networks can extend beyond the physical area they are meant to cover.

Figure 1 shows an example of when this can be a problem. In the figure, the dotted circle represents the physical range of the access point placed inside of a building. The area enclosed in this circle is the *coverage area* of the network. Sometimes, however, the owner of the network would like to impose an artificial limit on the wireless signal, restricting access to clients within a smaller physical area. In Figure 1, this area is the white square within the shaded circle, and is known as the *service area* of the network.

In this scenario, the coffee shop owners would like to limit network access to their customers in order to provide the best possible performance and security. In particular, nodes A and B should be granted network access, since they are withing the building, but nodes C-E, even though they are within the coverage area, they are outside of the desired service, and should thus be refused network access. Such a mechanism would also be desirable in an apartment complex, where a resident may want to prevent people in neighboring apartments from accessing the network and using the Internet connection that the resident has paid for.

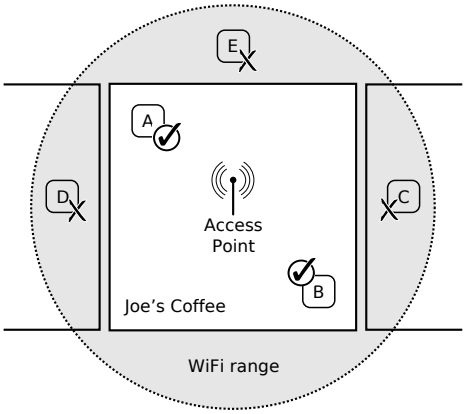


Figure 1. The system should allow WiFi access to hosts A and B, but prevent hosts C-E from accessing the network, even though they are within wireless range.

Unfortunately, network operators do not currently have a way to define the service area of the network, and are instead forced to provide wireless service to the entire coverage area. While some environments (such as a private apartment) can use link-layer access control methods such as WPA or WEP to limit which clients can access the network, these approaches are cumbersome to deploy in semi-public networks such as those found in coffee shops. Other methods for WAP access control, such as SSID hiding and MAC address filtering, face similar usability issues.

It is important to note that different types of access control methods target different network issues; some may be concerned with simply restricting access to authorized users, whereas others may aim to provide data confidentiality and integrity as well. WPA authentication [4] is an

example of the latter, and the Boomerang system is an example of the former. It should be noted, however, that Boomerang can easily be composed with methods such as WPA authentication to provide even strong access control or security guarantees.

1.2 Goals

The goal of our project was to develop a method that allows wireless network administrators to define the wireless service area for a WAP based on the acoustic fingerprint of the desired service area. We chose to use audio as the basis for our system because most wireless devices today, such as laptops and cell phones, have a built-in microphone for audio input and because the environments where our system is intended to be used often have music or ambient background noise that could be used as a reference signal.

In addition to the primary goal of being easy to deploy, we aimed to satisfy the following additional goals in designing and implementing our system:

Access Restriction The system should actually prevent users who are outside of the service area from consuming network resources; other security concerns such as confidentiality and integrity are secondary, and can be obtained by using existing systems such as WPA in addition to our system.

Flexibility The access control mechanism should work in environments with different audio properties, such as noisy public places like coffee shops, as well as quieter places like a residential apartment.

Portability The system should work with a variety of different client devices, operating systems, and web browsers.

The rest of this paper is organized as follows. Section 2 describes the architecture of the system, Section 3 discusses the implementation details, Section 4 evaluates the our system in two different environments, Section 5 presents some directions for future work and Section 6 concludes.

2. Architecture

The Boomerang system consists of three main components: 1) a *client* program that runs on wireless nodes wishing to access the external network; 2) a *verification* process running on a centralized server that performs the client authentication; and 3) a *gateway* process that restricts access to the external network to authorized clients only.

2.1 Design Approach

As described above, we chose to use sound as the basis for our new access control mechanism because most wireless devices today have a built-in microphone. To match the coffee shop use case described above, our first design relied on using the background music usually played in these establishments as a carrier for transmitting a network key (i.e. a WPA pre-shared key) to the client devices.

This approach would have provided link-layer access control, restricting access to both the local network and the external network connection to only those nodes within acoustic range that could receive the network key. By encoding the network key in frequencies outside of the human hearing range, we could "play" the network key on top of the music without sacrificing the music's quality. However, we discarded this idea due to the low quality and limited frequency response of the built-in microphones on consumer-grade devices today. We thought that most consumer-grade microphones would simply be unable to receive the signal in the frequencies we wanted to use as a carrier.

Our second design—and the design that our current system is based on—relies on a robust acoustic fingerprinting algorithm (described in Section 3.3) to match an audio sample recorded by the client to a separate *reference* sample recorded by a centralized server. As long as the client and server samples match, the client will be allowed network access. For this design to work, however, the client needs to be able to communicate with the authentication server over the wireless network *before* the matching can be performed. Thus, this design needs to provide access control at the the network layer, rather than the link-layer, as the previous design allowed for.

If providing link-layer access control was important, an alternative design would have been to have a separate *authentication network*, where unauthenticated clients first connect to communicate with the authentication server; after successfully verifying the client sample, the server would then allow the client to associate with a secondary, private network, where only authorized clients could connect. At least one commercial product uses a similar two-network approach to provide better security for wireless hotspots [2]; however, we felt this would be too cumbersome for users, since it requires the clients to manually switch to a new wireless network after authentication.

2.2 Client Process

The client process is an application that runs on wireless nodes wishing to use the network. By default, all clients

are prevented from accessing the external network and their access to the local network is restricted to the verification server. To gain unrestricted network access, clients need to successfully complete the acoustic authentication protocol described below. A discussion of how wireless nodes obtain the client application is deferred until Section 3.1.

The client begins the authentication protocol by using its local microphone device to capture an audio sample of the general ambient sound at its location. Once the local capture is complete, the client process connects to the authentication server and uploads its local sample. The server will then perform the acoustic verification algorithm (Section 3.3) and inform the client whether or not the authentication was successful. Authenticated clients must periodically re-authenticate with the verification server, to ensure that the clients stay within acoustic range of the access point for the duration of their wireless session.

2.3 Verification Process

The Verification Process (VP) is responsible for authenticating wireless clients, and keeping the gateway process informed of which clients are authorized for network access. Because of the computation complexity of the verification algorithm, the VP is executed on a dedicated server separate from the gateway.

To perform the verification, the VP must receive an audio sample from the client and compare the client's sample to its own sample; if the samples are determined to be a match (as described in Section 3.3), the VP updates the gateway process, informing it that a new client has successfully authenticated and should be given network access.

2.4 Gateway

The Gateway Process (GP) runs on the WAP and enforces access control for the network. Its goal is to prevent unauthenticated clients from sending or receiving traffic to or from any host other than the verification server. To keep track of which clients are authorized for network access, the GP maintains a *whitelist* containing the IP addresses of clients that have successfully completed the acoustic authentication protocol. All authenticated clients are authorized for unrestricted access to the local and external networks. All packets from unauthorized clients to hosts other than the verification server are dropped by the gateway.

3. Implementation

This section presents our implementation of the Boomerang WAP access control scheme outlined in Section 2.

3.1 Client Redirection and Access Control

When new wireless clients first connect to the access point, Boomerang uses DNS redirection techniques to implement a *captive portal*, intercepting all outgoing web requests and redirecting users to an internal webpage on the verification server. This internal webpage provides the user with information about the Boomerang system, and instructs the clients on how to download and run the authentication program. Until the wireless client runs the program and successfully completes the authentication protocol, all other non-web traffic is silently dropped by the gateway process.

Upon completion of the authentication protocol, the server will provide the client with an *authentication ticket* that contains a time-to-live value (TTL) for the user's credentials, indicating how long the user's credentials are valid. Once this TTL has expired, the client must repeat the authentication protocol to obtain a new authentication ticket. Thus, to maintain network access for the duration of the session, the client should repeat the authentication protocol before the TTL expires, requiring clients to remain within acoustic range of the access point for the duration of their session.

Boomerang enforces wireless access control by using a Linux-based WAP and iptables [8] to track the whitelist of authorized clients. Modification of the whitelist is performed by the verification server, which establishes a connection to the WAP and sends control messages to add or remove client IP addresses from the list as clients enter and leave the network.

3.2 Capturing Audio

Both the client process and the verification process are implemented in C and use the Advanced Linux Sound Architecture (ALSA) API [1] to record samples from the local microphone. The client process begins authentication by capturing a short sample (typically 10 seconds) from its microphone to a temporary file on the local disk. The client then establishes a connection to the authentication server and transmits this file to the VP for verification.

Because client authentication requests can come in at any time, the VP maintains a *sample window*, which is a fixed-size circular audio buffer that contains the most recent n seconds of sound recorded by the VP, where n is typically 10-20 seconds. A client's sample can match any portion of this sample window and thus, the size of this window determines how stale a client audio sample can be while still being considered a possible match. For

robustness to network delays, the sample window should be slightly longer than the client sample length.

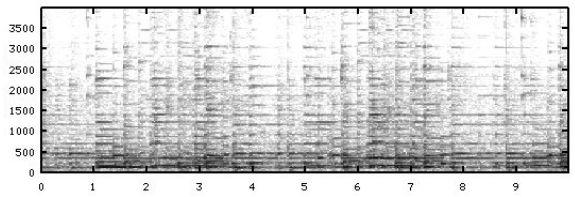
When a new incoming client authentication request arrives, the VP will take a snapshot of the circular buffer and pass the snapshot and the client's sample to the comparison algorithm. If the comparison algorithm determines that the client sample *matches* the server sample, then the VP will tell the gateway process to add the client to the whitelist.

3.3 Acoustic Fingerprinting

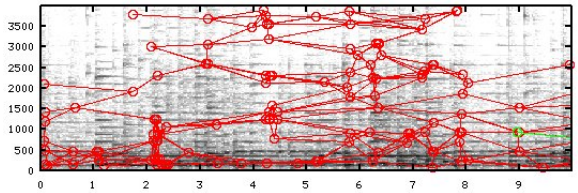
The comparison algorithm that Boomerang uses for matching the client and server samples is based on the algorithm used by the popular Shazam music identification service [7]. The choice of this algorithm was due to its robustness to noise and environmental effects (such as reverb or ambient sounds) in the signals being compared. The algorithm was designed to compare client samples taken from mobile phone microphones to a database of known acoustic fingerprints. Because phones are often used in noisy environments and the quality of the mobile phone's microphone is often poor, the algorithm was designed from the ground up to tolerate these variations in the signal. For a full description of the Shazam algorithm, we refer the reader to [6]; we provide a brief high-level description of the algorithm here.

The Shazam algorithm relies on the presence of a *reference database*. The reference database contains a compact representation of the acoustic fingerprints for one or more known *reference samples*. Given a *query sample*, the algorithm will compute the query sample's acoustic fingerprint, and compare the result to those contained in the reference database. If a match is found, the ID of the matching sample is provided, along with a quantification of how well the samples matched. The Boomerang system constructs a new reference database on each incoming client request, containing just the reference sample provided by the VP; the client sample is then used as the query sample in the Shazam algorithm. An extension to this algorithm that uses multiple server reference samples is discussed in Section 5.2.

The same acoustic fingerprinting algorithm is used for both the client and server samples. The algorithm begins by computing a spectrogram of the sample by performing a Fast Fourier Transform of overlapping 64ms periods. An example of a computed spectrogram is shown in Figure 2(a). In this image, the vertical axis represents frequency, the horizontal axis is time, and darker areas represent higher amplitudes.



(a) Spectrogram only



(b) Spectrogram with landmarks and landmark pairs

Figure 2. An example spectrogram and landmark computation of the acoustic fingerprinting algorithm.

Given this spectrogram, the algorithm then finds local maxima to use as *landmarks*. These landmarks are then used as *anchor points* to create pairs of landmarks close together in time, establishing a temporal relationship. Next, the landmark pairs are translated into a hashed identifier and placed in a table for quick lookup. An image of what the spectrogram with landmarks and landmark-pairs highlighted is shown in Figure 2(b). The circles show the landmarks, and landmarks that are connected by an edge indicate a landmark pair.

Once the VP has computed the fingerprint for the server's reference sample, the process is repeated on the client sample. A "sliding window" comparison is then used to search for the temporal offset of the client sample, relative to the beginning of the server sample. The offset that is chosen is the one that produces the highest number of matching landmark pairs. The absolute number of matching landmarks is used to quantify the quality of the match.

Our system uses an open-source implementation of this algorithm written in MATLAB by Dan Ellis [5].

4. Evaluation

4.1 Methodology

The equipment that we used for our experiments consisted of a netbook with a built-in microphone as the client, a desktop PC with a 3.0 GHz dual-core CPU as the verification server, and a Linux-based Linksys WRT54GL wireless access point and gateway.

The primary metric that we used to evaluate our system was the number of landmarks in the server and client sam-

ples that matched in a given trial. Because the comparison algorithm is very unlikely to find matching landmarks in samples that are known to be different or random, a low threshold can be used for the number of matching landmarks needed to declare a match. The original MATLAB implementation suggests a threshold of 5 or more matching landmarks to qualify as a match, and we found this to be an appropriate value for our tests. Because users will be unlikely to adopt a new access control mechanism if it takes a long time to authenticate, we also measured the time required for a client to run the authentication program. All data points that we present are an average of at least three trials.

We evaluated these metrics in two environments: a quiet room with no ambient noise to simulate an apartment, and a busy coffee shop with a moderate amount of ambient noise. Music was played in each environment at approximately the same volume.

In each environment, the primary independent variable was the distance between the microphone used to capture the server's reference sample and the microphone used to capture the client sample: by tracking the number of matched landmarks as this distance changes, we gained an understanding of how well the Boomerang system would work in practice as an access control mechanism for WAPs with clients attempting to access the WAP from various locations.

Audio samples were captured on the server with an external microphone; samples on the client side were captured with the microphone built-in to the netbook. We experimented with using an external microphone with the netbook client as well, but found that it did not improve the matching performance. Further, we felt using the built-in microphone was a more realistic test case since it didn't rely on wireless clients having any additional hardware.

Before running our tests, we spent some time tuning some of the parameters used by Boomerang during the acoustic fingerprinting algorithm. Modifying these parameters changed the number of landmarks found in the client and server samples. There are a number of parameters that offer a trade-off between accuracy and computation time. When the number of landmarks used for matching is increased, making the comparison more accurate, the comparison takes longer to perform; conversely, reducing the number of landmarks found reduces computation time, but will be less likely to find a match between the client and server samples. In all tests, we used 10 second audio samples from both the server and the client, with the remaining parameters set to return as many landmarks as possible while maintaining a reasonable execution time.

4.2 Apartment environment

To determine the limitations of our system in the optimal case, we first tested it in a room with background music and no ambient noise, which matches our proposed use case for an apartment. We played the song at moderate volume on a set of speakers connected to the server, and placed the server's microphone directly next to the speakers to ensure a clean recording. We then performed the client authentication process at increasing distances from the server's microphone.

Figure 3 shows the number of matching landmarks between the client and server samples as the distance between the two microphones increased. Inside the room, up to 24 feet from the speakers, the number of matching landmarks easily exceeded the typical threshold of 5 landmarks used to determine if the samples are a match. The data points for 32 and 40 feet were actually obtained with the client netbook in the hallway outside the room. Even at 32 feet, however, the average number of matching landmarks still exceeded the threshold of 5, demonstrating the effectiveness of Boomerang even at great distances in this environment.

The figure clearly demonstrates an inverse relationship between the number of matched landmarks and the distance between the two microphones. This shows the importance of the volume of the sample captured by the client in the matching process.

In order for the fingerprinting algorithm to pick out appropriate landmarks from the audio sample, the volume of the "signal" (the music, in this case) must be greater than that of the "noise." In this test, because there is no ambient noise in the environment, the only noise captured by the client sample is an artifact of the hardware and is minimal. Therefore, even when the music captured by the client is barely audible, it still stands out in the client's spectrogram, allowing the fingerprinting algorithm to find enough of the right landmarks to determine that there is a match.

Fingerprint matching time During our experiments in the quiet apartment environment, we measured the total elapsed time from when the client executed the authentication program to when the client received the server's response. At all distances, the total matching time averaged between 30 and 35 seconds; this time was dominated by the sampling time (fixed at 10 seconds) and the landmark finding and matching time. The time spent transmitting the client sample to the verification server was negligible compared to

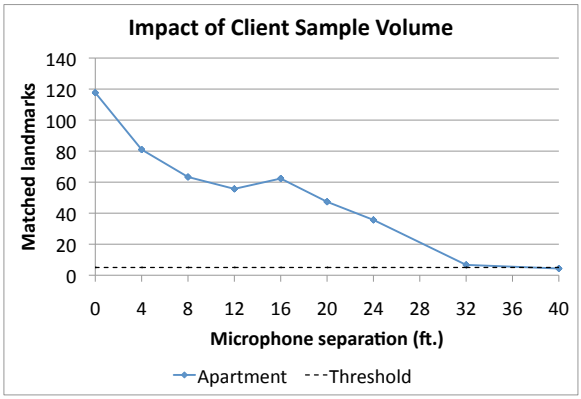


Figure 3. Matching performance in an apartment environment with a high-quality server sample and no ambient noise. In this optimal environment, matching succeeds even if the background music is just barely audible to the client.

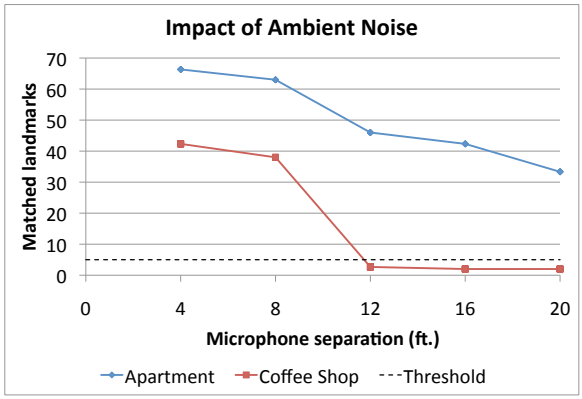


Figure 4. Comparison of apartment and coffee shop environments. Ambient noise in the coffee shop leads to fewer matched landmarks, causing authentication to fail when the client microphone is more than 8 feet away from the server microphone.

these other factors, and thus the total matching time did not significantly increase as the client got further away from the access point, as might be expected due to the decreased wireless signal strength.

4.3 Coffee shop environment

Our second set of experiments were conducted in a coffee shop. The coffee shop had music playing from several speakers around the room and had a moderate level of ambient noise due to customer conversations and espresso machine noise. The server's microphone was placed about four feet away from one of the speakers in the room.

Figure 4 shows the number of matching landmark pairs between the client and server samples in the coffee shop, and the number of matching pairs in a similarly configured scenario in a quiet apartment. The purpose of this test was to understand the effect that ambient noise in the environment had on the ability of our algorithm to match the audio samples. The graph shows the number of matching landmarks for each environment as the client netbook is moved further away from the primary audio source. The results of these tests demonstrate that the level of ambient noise in the environment has a significant impact on the ability of Boomerang to correctly identify matching audio samples.

One explanation for this is due to the way landmarks are chosen in the audio samples. Because each microphone will pick up the background noise and conversations of the people around it, ambient noise tends to be a localized phenomenon, with different physical locations experiencing different background noises. More ambient noise generally decreases the signal-to-noise ratio of the audio samples used for matching, so as a result, some of the landmarks chosen by the fingerprinting algorithm may be taken from the localized noise, rather than the music signal.

This localization of noise tends not to be a problem when the client and server microphones are close together, as evident by the large number of matching landmarks at 4 and 8 feet of separation. Because both samples experience the same ambient sounds, even if the landmarks are chosen from the background noise instead of the music, there is still a high probability of finding a match. However, as the microphones move further apart, the localization of ambient sound tends to negatively impact the matching ability of Boomerang. As shown in Figure 4, we were only able to authenticate at a range of 8 feet in the coffee shop, whereas the apartment setting was able to authenticate beyond 20 feet away.

Figure 5 shows the effect on the matching ability of Boomerang for different levels of quality in the server reference sample. The high SNR samples were taken with the server's microphone aimed directly at the speaker at a distance of about 4 feet, resulting in reference samples in which the music signal was louder than the surrounding ambient noise. The low SNR samples were captured with the server's microphone sitting on a table about 8 feet away from the speaker, pointed into open space; this resulted in samples with no distinct music signal, but rather a fairly constant level of background noise. While we did not quantify the SNR of the samples we captured, we did qualitatively verify the SNR properties of the samples by playing them back after capture.

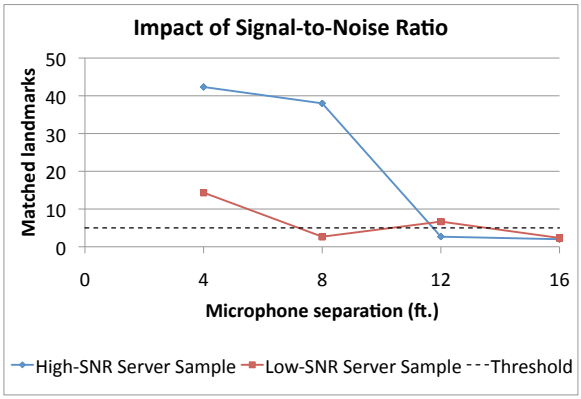


Figure 5. Comparison of different quality reference samples. The sample with a more audible music presence had more matching landmarks at 4 and 8 feet than the lower quality sample. However, both samples matched poorly beyond 8 feet of separation.

The results in Figure 5 show that Boomerang's matching ability is greatly improved by having a server sample with a high signal-to-noise ratio. As explained above, a high-quality server sample results in high-quality landmarks being isolated by the fingerprinting algorithm; that is, with a strong music presence, most of the landmarks chosen by the algorithm will come from the actual music being played, rather than the environmental noise. This tends to focus the matching algorithm on these high-quality landmarks when comparing to the client sample. Unfortunately, this means that for our coffee shop use case, just matching client samples to the ambient noise will not work well, and some strong audio signal must be present.

Because we were unable to control the wireless infrastructure at the coffee shop, our tests relied on a verification server that was accessible over the Internet, rather than one that was on the local network, as our design suggests. As a result, we do not present detailed timing results for these experiments, but can qualitatively assert that the authentication latency was dominated by the time spent transmitting both the client and server samples remotely to the verification server, and not with the actual fingerprinting analysis.

4.4 Evaluation Summary

Our evaluation shows that Boomerang works very well when there is little ambient noise in the background, and that the sample matching algorithm tends to find better matches when at least one of the samples is of high quality. However, even in the coffee shop environment, where there was a relatively large amount of environmental noises, our system worked well enough to authenticate clients up to 8 feet away when using a high quality server sample. Further, our first test showed that total end-to-end authentication time was on the order of 30 seconds, a duration we feel is quite reasonable for our system.

5. Observations

5.1 Lessons Learned

One of the most challenging aspects of this project was developing a system that was relatively portable across multiple clients. There are a number of different operating systems, each with their own audio API, and the client program we wrote in C for Linux did not port directly to other systems. In Section 5.2, however, we discuss some ideas for how we can develop a more portable client application.

We also discovered that working with microphones on client devices is problematic due to the differences in client audio configurations. The samples we captured were very sensitive to the numerous levels in the software mixers that can be adjusted in each client's audio control panel, including things such as the microphone boost level and the sound card capture level. For our experiments, we adjusted these levels until we found settings that produced qualitatively "good" audio samples (i.e., the highest level of audio that captured without distortion), then fixed these settings for all of the tests. Handling these settings in a real deployment with varied client devices would present a significant challenge, and would almost certainly require the client program to employ an automated level setting mechanism to dynamically adjust the client's capture level.

Finally, we found that the built-in microphones on our client devices were of poor quality and introduced moderate amounts of noise in to the captured samples. This negatively impacted the quality of landmarks found by the fingerprinting algorithm, reducing the effectiveness of the Boomerang system. Systems that rely on client hardware devices should expect and be built to tolerate the limitations of these devices.

5.2 Future Work

Our current Boomerang implementation and its limitations suggest several directions for future work. A few of these possibilities are discussed below.

Improved Client Interface Rather than distributing the client program as a downloadable file from the authentication server, we would like to implement it as a web application that runs in the client's web browser when they are redirected to the server for authentication. This would not only be easier for clients to use, but it would be much more

portable across operating systems. Although accessing a device's microphone through standard web browser programming techniques is not currently feasible, some plugins such as Silverlight offer limited support for microphone device access [3], and the emerging HTML5 standard may include microphone support as well.

Multiple Reference Samples One of the weaknesses of the Boomerang system is its reliance on a single reference sample captured from the server's microphone. A system that captures multiple reference samples from multiple microphones distributed around the desired WiFi service area would offer improved matching ability for two reasons. First, using multiple server microphones throughout one room means that the distance between the client microphone and the server microphone would likely be reduced, thereby increasing the number of landmarks that are found and matched. Second, the system would also work in a house or a coffee shop with multiple rooms, all within wireless range of the access point, since one microphone could be placed in each room to capture the audio particular to that room.

Improved Security Our system focuses on the use of access control to prevent resource consumption by unauthorized users, and does not directly address other security concerns, such as confidentiality and integrity. However, because Boomerang operates at the network layer of the OSI model and common encryption techniques such as WPA and WEP operate at the link-layer, WPA and WEP are directly composable with Boomerang and can be used in addition to Boomerang to provide additional security guarantees. For example, WPA can provide encrypted communication between the WAP and wireless clients, and Boomerang can further restrict access to the nodes that are within a certain area.

6. Conclusions

We have developed Boomerang, a system for network access control based on acoustic fingerprinting techniques. Boomerang utilizes the built-in microphones found on most wireless devices today to perform client authentication without requiring users to type in passwords or manually update access lists. Our evaluation of a prototype implementation showed that the system consistently authenticates clients in environments with a strong audio signal that stands out from any noise. The prototype did not work as well in a coffee shop with moderate ambient noise, but our directions for future work suggest methods for improving acoustic fingerprint matching in noisy environments. By incorporating these promising techniques, we believe that Boomerang could be developed into a viable method for WAP access control.

References

[1] Advanced Linux Sound Architecture (ALSA). `http://www.alsa-project.org/main/index.php/Main_Page`.

[2] AeONsafe - How AeONsafe Secures Your Wireless, October 2006. `http://www.aeonsafe.com/ShowArticle.aspx?ID=64`.

[3] The Official Microsoft Silverlight Site, November 2009. `http://www.silverlight.net/learn/videos/all/access-web-camera-microphon%e/`.

[4] Wi-Fi Alliance. The state of Wi-Fi security. White paper, September 2009. `http://www.wi-fi.org/knowledge_center_overview.php`.

[5] Dan Ellis. MATLAB Implementation of the Shazam Music Identification Algorithm, May 2009. `http://labrosa.ee.columbia.edu/matlab/fingerprint/`.

[6] Avery Wang. An industrial strength audio search algorithm. In *ISMIR*, 2003.

[7] Avery Wang. The Shazam music recognition service. *Commun. ACM*, 49(8):44–48, 2006.

[8] Harald Welte. netfilter/iptables FAQ, October 2003. `http://www.netfilter.org/documentation/FAQ/netfilter-faq.html`.