Securing Wearables through the Creation of a Personal Fog

Charles Walter
The University of Tulsa
charlie-walter@utulsa.edu

Ian Riley
The University of Tulsa
ian-riley@utulsa.edu

Rose F. Gamble
The University of Tulsa
gamble@utulsa.edu

Abstract

Increased reliance on wearables using Bluetooth requires additional security and privacy measures to protect these devices and personal data, regardless of device vendor. Most wearables lack the ability to monitor their communication connections and protect personal data without assistance. Attackers can force wearables to disconnect from base stations. When a wearable loses its connection to its base station, an attacker can connect to the wearable to steal stored personal data or await reconnection to the base station to eavesdrop on communications. If the base station inadvertently disconnects from the cloud serving a security-aware app, it would be unable to respond to a rapid change in the security of its current environment. We design a personal fog incorporating wearables, a base station, and the cloud that allows the wearable to be situationally aware and manage inter- and intra-fog communications, given local personal fogs with the same app.

1. Introduction

Bluetooth devices, such as wearables, have become ubiquitous in day-to-day life. With this ubiquity comes a rush to get devices to market quickly, often at the cost of good security and privacy standards. Many of the devices currently available use the basic built-in Bluetooth security. While this has been improved over each new version of the Bluetooth security standard, there are many known holes in the previous and current standard [7] to necessitate a method of preventing an attacker from gaining personal information from user devices. Additionally, with an Ubertooth One [8], an open-source hardware device capable of intercepting Bluetooth transmission, and crackle [2], an open-sourced software program which can decrypt Bluetooth communication packets, an attacker can gain personal data by intercepting device communications using the built-in Bluetooth standards.

Without additional security features, many devices are open to interference from malicious users. For example, an attacker can force the disconnection of a wearable from its base station. Once disconnected, the attacker can intercept the pairing packets and gain a user's personal information without any alert to the user. This method can be exploited to great effect if the wearable is transmitting very sensitive data. If wearables rely only on default Bluetooth standards, they won't be equipped to prevent attackers from using this exploit. More advanced wearables, such as the Apple Watch 2, can operate for some period without connection to their base station. As these advancements proliferate, they open additional attack vectors for disconnected, but still operable, wearables.

If a base station connected to a wearable has a method of recognizing its environments' insecurity, it would still have major issues in practical use. First, it would need to rely heavily on users informing the application that they are unsafe in their current environment. This makes this solution far less effective for users that do not have the expertise to properly evaluate the risk present in their current environment.

However, users without the proper expertise could rely on cloud-computed rules that dictate which environments are unsafe. These rules could be computed from data that is provided to our application by users who are particularly security conscious. Even with enough security conscious users, there are no guarantees that users would be able to receive updates to their application to utilize the new rules being created as there are many situations where users would have poor-to-no cloud-connectivity. In environments where users cannot receive updated rules it is possible that they would be transmitting personal data even though their environment had already been deemed unsafe by the cloud. Finally, as most current wearables are unable to differentiate if a connected device is truly secure, any information stored on the device could be requested by an attacker who connected to a disconnected wearable.

To increase the security of these devices, we need wearables that are capable of better managing their own Bluetooth connections without requiring direct user involvement. Relieving the security and

URI: http://hdl.handle.net/10125/50583
ISBN: 978-0-9981331-1-9

situational awareness burden from the user requires management facilities such as knowledge of the devices to which it is currently connected, the ability to establish new connections to other devices that are not its base station, refuse incoming connections, and terminate established connections. This additional computing power on the edge (i.e. at the wearable) would allow the introduction of these facilities to increase personal security and privacy of wearable operation and data.

Even with this added power at the edge, current consumer wearables require unique APIs for use with each device, requiring developers to create applications with different code when a new device is introduced. This makes it difficult to develop a generic solution which will work on multiple devices, and necessitates looking at the direction wearables are moving.

In this paper, we introduce a concept of a personal fog computing system as part of the quantified self. It requires elevating a wearable to a computational device, which has already been shown to be feasible with the Apple Watch 2. Adding computational power to the edge via the wearable, a personal fog introduces wearable autonomy for security protection. This architecture contrasts with most of today's wearables, which solely rely on their base station.

A personal fog has additional benefits. Situational awareness can be enhanced by creating a security app, downloadable by a wearable, where nearby personal fog networks can detect app advertisement and inform another's wearable when a social environment is insecure. If a wearable becomes disconnected from its base station, either inadvertently or maliciously, we describe a method for temporarily fostering wearable devices in nearby personal fogs that share the security app. Fostering, in this scenario, means sending a single packet to "quiet" a rogue wearable in an insecure environment until it can reconnect to its base station. The app also allows the home base station to quiet its wearables. We demonstrate how this solution works by implementing it in a small-scale testing environment and simulating a large-scale application. We analyze our application as it relates to the CIA triad of confidentiality, integrity, and availability.

2. Background

Kattepur et al. [6] examined the use of fog computing to improve battery life and network communication speed of robots performing a computationally intensive task. Using the fog, they improved latency of communication by 77% when sorting information and they achieved a 54% improvement to battery life. This method shows the improvements that are possible even with similar processing power at each level of the fog, as the same robots were used for all fog layers. They did not test their solution in the real world, however, so it is possible issues which could not be simulated may occur when attempting to use this method.

Hong et al. [5] describe a generic fog system allowing custom code to be loaded into general fog machines based on their location via a simple appkey. This system allows developers to deploy their systems to fog devices which are already in the locations they are targeting. This system would allow our solution to adapt to security threats more quickly. However, their only test of their system was in simulation.

Vaquero and Rodero-Merino [9] provide a useful definition of what the fog really is. Their discussion on what it means to be a fog is vital to the development of the fog. They settled on a model needing ubiquitous devices communicating with each other to perform storage and processing automatically, rewarding users who allow their devices to be part of this system. Yi et al. [12] provided a survey of fog computing concepts, applications, and potential issues in design and implementation of a fog computing system based on this definition.

Giang et al. [4] designed a flexible fog computing model for use with VANETs. Because VANETs suffer from high latency, using the fog by including processing on the edge would limit the amount of data which needs to be transmitted and, because of the proximity to processing nodes, lowers the latency of the communication between car and network. This paper does not address the issue with a network where nodes drop in and out.

There has been work on wearable devices in situations which may require our automatic method of quieting devices. Abie and Balasingham [1] focused on the creation of a framework for Internet of Things devices in the healthcare industry. Their system focused on wearable sensors communicating with a smartphone which would pass this data on to a healthcare professional. This system would be a prime target for attackers wishing to steal personal data and would likely include many users who are not conscious of Bluetooth security practices. This type of system directly motivated this work.

3. Prior efforts

3.1. Original solution

Previously, we created an application specific to the Apple iPhone that adapted the state of its Bluetooth communications based on the perceived security of its

current environment [10]. The application collects as much data as it can from the iPhone and its connected wearables to determine if it is in an insecure environment. Once informed of a potential threat, the iPhone maintains a connection to the wearable but does not send additional information or request information from the wearable until it detected or is informed it is in a secure environment. Maintaining the connection without information sharing requires the sending of exclusively empty packets. An example is shown in Figure 1. In this case, the phone is in an insecure environment only for communication with a Bluetooth headset and was considered secure if the headset was not connected, as seen in Figure 1a. Once the headset was connected however, the phone recognized the insecure environment and stopped sending data to the now connected Bluetooth headset, as seen in Figure 1b.

This method is viable primarily because wearables contain storage on the device to collect data when a device is disconnected from its base station. Thus, the application can stop sending data without losing any important information the wearable might still collect. Different wearables have different amounts of storage for their data, but most can store more than two days' worth of data before overwriting.

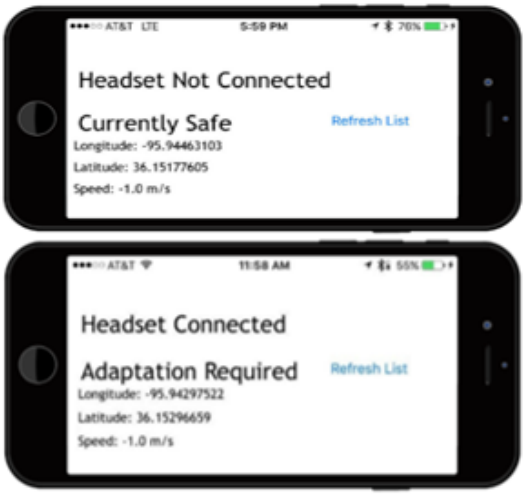

Figure 1. iPhone application a) before adaptation and b) after adaptation

There were issues with this method. First, we chose to work with the Apple iPhone, as it is seen by developers as the most restrictive platform for app development and it is widely used by consumers. Porting the application to less restrictive OSs proved difficult. For example, the Android framework does not allow for specific knowledge of connected Bluetooth devices at runtime, requiring workarounds to learn what devices are connected.

3.2. Adding the cloud

To provide a more variable ruleset for wearable adaptation while in an insecure environment and introduce crowd-sourcing to improve and hasten wearable knowledge regarding such environments, we extended our app to use cloud-based machine learning algorithms to adjust to newly-forming insecure environments [11]. This method allowed the app to operate regardless of connection to the cloud, as all rules would be stored on the device and updated when a connection could be reestablished.

The app took a snapshot of its current state at regular intervals. Snapshots included the connected devices, all information available to the phone, either from the wearable (e.g. heartrate) or from the phone itself (e.g. time), and the current security state of the environment. The snapshots were passed to our cloud-based machine learning algorithm, which would use existing snapshots to learn to identify potential insecure environments. The results of this learning were then translated into rules for use in our app.

Our approach stored all rules on a single cloud service which was queried periodically to see if the current environment was secure. This method was accurate when predicting if an environment was insecure and, with the low risk of Wi-Fi encryption being broken for secure communication, making it a good option for predicting if the phones current environment is insecure.

While the use of a cloud-based machine learning algorithm improved the speed at which newly insecure environments were discovered, it still allowed for a window of time where devices were susceptible. For example, if an attacker is sniffing Bluetooth traffic in a coffee shop, the cloud would not be able to truly adapt to this problem quickly. It would only allow those who told the app they were insecure to be secure until the cloud was able to "catch up" and tell the other users. Thus, an attacker can stay in one place and collect information from users who believe they are secure until the cloud is able to recognize the newly insecure environment. Once discovered, the attacker is free to move to a new location which is marked by the cloud as secure and start eavesdropping on the communications there.

Another issue with this extension was the inability of our app to work for all Bluetooth devices. Because many Bluetooth device manufacturers create custom APIs for interacting with their devices, we were forced to create custom code for each device tested. While it was possible to prevent communication with these devices, the lack of a generalized Bluetooth API prevented us from creating an app that would work with every Bluetooth device currently on the market.

3.3. Addressing confidentiality, integrity, and availability

Our previous app focused on the confidentiality and integrity of wearable data by preventing data which may be intercepted from being transmitted in insecure environments. In both our original solution and the improved cloud-based solution, we maintained the focus on confidentiality and integrity of the data by protecting what was streamed from a wearable device to its base station. The drawback is that the data is not immediately available when the wearables are forced to send empty packets.

4. Designing the personal fog

4.1. Increasing wearable computational power

Informing users of a potentially insecure environment should be done as rapidly as possible, especially when they are trusting an app to have this knowledge. Relying solely on a cloud service based on crowd sourcing means such an app would assume it was secure until enough users determined they were insecure for our cloud service to learn of a new insecure environment. This type of service also requires large numbers of users to be engaged in their environment and recognize potential insecurities. In addition, it requires constant connectivity to obtain information from the cloud and to prevent an attacker from accessing data stored on a "quieted" wearable.

To address these pitfalls while retaining the benefits of the prior work, we extend our app, including the cloud service usage, to a new computation and communication model. Moving the traditional quantified-self architecture to a *personal fog* is illustrated in Figure 2. The difference in the models appears slight, but the personal fog capitalizes on edge computing to increase Bluetooth security and privacy.

The new model allows for rapid communication of insecure environments while focusing on the privacy of a user's personal identifying information, moving some of the burden of situational awareness from the user to the wearable. By increasing the computational power on the edge, the wearables can become aware of other wearables using the app, as well as their state if disconnected, either from the cloud or their base station. This increased computational power also allows the wearable to process data by checking against XML rules provided by the app and stored on the device based on the sensor capabilities of the wearable. For example, if a wearable has access to heartrate sensors, any rules related to heartrate will be stored on the device and checked to ensure that, should the wearable be in an insecure environment, it will not transmit any personal information which may lead to the user of the wearable being identified.

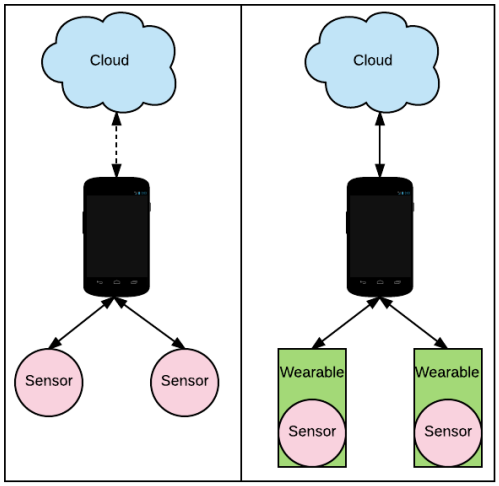

Figure 2. Left - current quantified-self model Right - personal fog with wearables model

Where the traditional quantified-self has its edge devices as the phones, wearable edge devices with increased computing power can now process the data collected from their own sensors before sending that information to the phone. Unlike our previous work, this allows the wearable to package its snapshot information separately from the phone. The phone can perform additional data processing on the stream of information it is receiving, such as compressing the snapshots from the wearable device or checking device specific security rules, before sending to the cloud. As before, the cloud will return information on the predicted security of the phone and the wearables environment, along with learned security rules for the devices to follow. With that information, the phone can secure itself, as before. However, with increased wearable processing power, we can push security and privacy information directly to the wearables, allowing for finer grained adaptations at the edge.

Another benefit of the personal fog concept is the increased ability to ensure wearables can operate securely in insecure environments by incorporating wearable situational awareness and intelligence of their current security state. If wearables are connected to their base station when they enter an insecure environment, the method previously used works to control transmission. However, it poses a problem if the wearable inadvertently disconnects from its base station before entering an insecure environment. In this case, the wearable would have no knowledge of its environmental state and could allow an attacker to connect and request data directly from the device.

Without information about its current state, the disconnected, and more powerful, wearable could not stop its communication. Our approach to resolve this issue is to enable wearables to communicate with valid base stations external to their initial fog, while still maintaining privacy and adapting without divulging any personal information. We label this interaction as *fostering fog devices* and describe a design and implementation of how it can be performed.

4.2. Fostering devices between fogs

Similar to VANETs [4], we establish dynamic computational networks between personal fogs to perform the fostering. We provide no guarantees that specific connections will remain static or that the topology of our fog networks will remain the same over time. The objective is to implement a network that is flexible while remaining true to the fog computing paradigm. However, unlike many other applications of the fog, our specific problem domain can contain a multitude of fog networks each with unknown and non-rigid topologies.

We have designed a set of protocols for fostering that use nearby personal fog networks to better aid the security of our app users by allowing, not only temporary connections between either two nodes in the same personal fog network, but also between two nodes that do not share a personal fog network. The temporary connections made by fostering last only long enough to inform the disconnected node of a potentially insecure environment before disconnecting, ensuring no transfer of personally identifying information between the nodes.

Our communication protocols create temporary connections in one of three ways.

- Foster a disconnected wearable with nearby base station
- Foster a base station disconnected from the cloud with a nearby base station
- Allow two disconnected wearables in the same fog to directly communicate

4.2.1. Wearable fostering. To secure wearable devices that become disconnected from their base station, as seen in Figure 3, we have designed a protocol that would allow for connected personal fog networks to temporarily foster another network's disconnected wearable device. Through this protocol, base stations in each personal fog network can accept incoming connection requests to foster a disconnected wearable by advertising an app specific service known to all devices running our app.

When a wearable becomes disconnected from its base station, it first attempts to discover other base stations in the area advertising the app service. If a base station which is in an insecure environment receives a fostering request, it will inform the wearable of the state of the environment to which the wearable will respond by adapting its state. The devices will then disconnect.

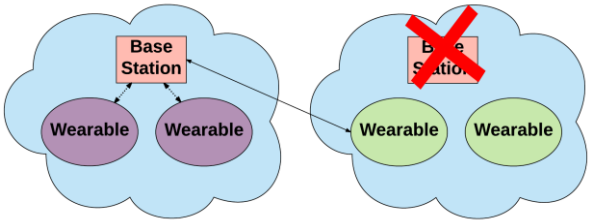

Figure 3. Wearable fostering. Left shows an unsafe personal fog (purple), right shows a disconnected fog device (green) being fostered

4.2.2. Base station fostering. If a base station is having trouble connecting to the cloud for new updates, it will attempt to establish a connection with the base station of another personal fog network in the vicinity, as seen in Figure 4. The disconnected base station attempts to discover nearby base stations advertising the app specific service. Once a connection has been established, the base stations can share with each other their knowledge about the environment and adapt their state, as well as the state of their personal fog. Such communication events could be triggered through one base station receiving an update from the cloud, from user input, or from another personal fog network if one of the networks were to have multiple external base station connections.

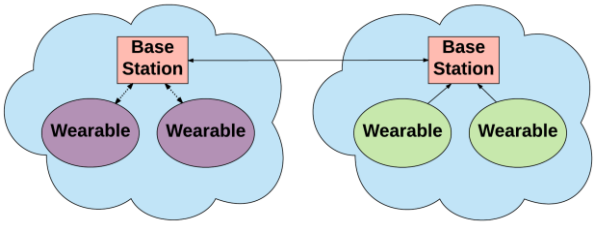

Figure 4. Fostering a base station. Left shows an unsafe personal fog network (purple) communicating this to an unaware network (green) on right

With this same method, certain base stations can have embedded app rules to be proactive and use the advertised service to alert other nearby personal fog networks that the environment is unsafe, for quicker adaptation. Base stations are not required to reflect the changes proposed by other personal fog networks, but

through this protocol each user can make more informed decisions about the risk present to their own wearable devices given the alerts received when using our app.

4.2.3. Intra-fog fostering. Sibling wearables in a single personal fog can connect to one another, as seen in Figure 5 to share security awareness. Typically, all wearables in a personal fog are connected to a base station within that network. These base stations are responsible for quieting the wearables in the network if the environment is deemed to be unsafe (as discussed in Section 3). However, if some of the wearables become disconnected from their base station then it is possible that they would not be properly quieted when the other wearables are, which can increase vulnerability to attack at both the wearable and the personal fog.

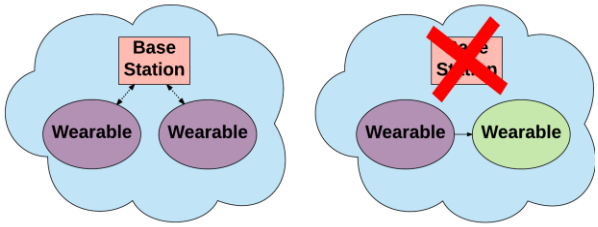

Figure 5. Intra-fog communication. Right shows a wearable (purple) informing its fog neighbor (green) it is unsafe

To prevent this situation, after a wearable has been quieted from a base station outside its personal fog, it subsequently attempts to establish a connection to its sibling wearable devices to inform them of the change in the environment. Wearable devices in each personal fog network advertise a unique service known only to members of that personal fog, restricting discovery. Once a wearable device has been quieted, it can attempt to discover other wearables which have not been quieted through this advertising. This communication technique is also useful if multiple wearables in the same fog become disconnected simultaneously and become fostered by different base stations. Following this protocol, disconnected siblings can work together to better secure the wearables of their own personal network.

4.3. Addressing confidentiality, integrity, and availability

An important thing to note about our system is that only wearable's original base station may inform it that it is safe again, because it has knowledge of properties unique to its personal fog, such as MAC addresses, a pre-shared unique ID of its personal fog, and a pre-shared unique key known only to the base station and the device itself. While some of this information, such as the MAC address and basic device information, is publicly available through a Bluetooth information request, the specific ID of the personal fog service and the pre-shared key should only be known to trusted devices and, being 128 bits long, are essentially not guessable by an attacker. We maintain the same level of confidentiality of our original app with the fostering approach. It increases the availability of our app data by dispersing the responsibility of informing base stations and wearable devices of an insecure environment. This way, should there be a disconnection from the cloud or a rapid change in the environment, our app is able to maintain its availability for local sharing. By restricting fostering to quieting a device, we ensure the integrity of our app data. It is not possible for an attacker to claim they are safe and prevent the app from recognizing an existing unsafe environment, as the app will continue to try to foster with other base stations in the area.

5. Implementation

To illustrate the communication protocol within a personal fog, we rely on the Bluetooth wearable testbed. This testbed, seen in Figure 6, is composed of three Raspberry Pi 3s, which can simulate currently available wearable devices using the Sense HAT add-on, USB connection of sensors, and built-in Bluetooth capabilities. The testbed provides extra processing power and Wi-Fi communication capabilities above current wearable technology. This additional processing power means a Raspberry Pi can be a client or a server at any given time, which allows the Raspberry Pis to simulate both the wearable edge and the base station within a personal fog. Using the Raspberry Pi as a simulated wearable, we avoid issues with proprietary APIs. By simulating the base station, we remove the need to port our application and experiments to different phones. The Raspberry Pi that is simulating the phone layer can communicate through Wi-Fi to the cloud, giving us a full simulation of each layer of our personal fog, as shown in Figure 2.

We implemented our original app (from Section 3) onto the Raspberry Pis, which migrated it to the personal fog, using devices B and C in Figure 6 to act as wearable devices connected to device A, which acts as the phone or base station. To implement our app, we used the Pybluez python library to handle the Bluetooth communication between the devices. After validating the app functionality on the hardware, we then forced a communication disruption with a

command from either the cloud or the phone, designating device C to represent either the base station or a wearable device that is disconnected, depending on the fostering solution we target. We assume from this point on that all devices are communicating from within our app using the same service ID and, when in the same personal fog, using a private personal fog service ID.

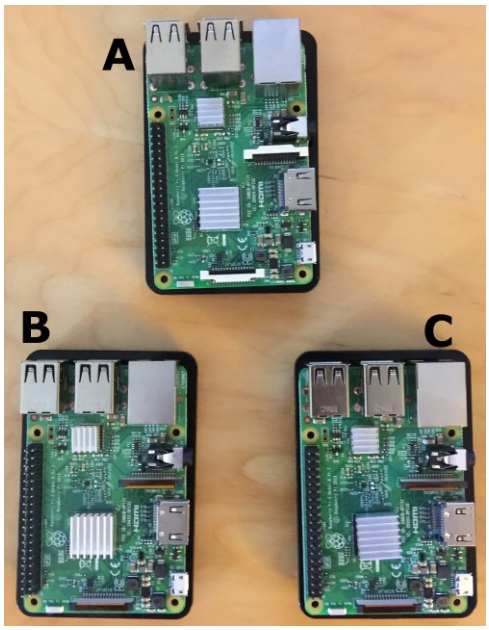

Figure 6. Three Raspberry Pi 3s used to test our solution

5.1. Fostering a wearable

As described in Section 4.2, the personal fog we construct allows for three types of fostering – wearable fostering by a base station, base station fostering by another base station, and wearable fostering by another wearable. We first examine wearable fostering by a base station. When a wearable (recall that it is a Raspberry Pi on the personal fog edge in our architecture) disconnects from its original base station, it can seek to be fostered by another base station (another Raspberry Pi) to discover the current environment's security state. It first sends an inquiry to all discoverable Bluetooth devices in range, asking for the services running on the devices. It may receive multiple responses, which when parsed identify valid base stations.

The wearable attempts to connect to the one of the valid base stations by sending a connection request as shown in the sequence diagram in Figure 7. If the base station is its original one, it will attempt connection with that first. If this reconnection is unsuccessful, it will choose another valid base station that responded. Because of the fluidity of the environment, it only waits 5 seconds for a base station response before choosing a new one to foster with. If all choices are exhausted, it waits 1 minute before inquiring again.

When a base station receives a connection request from a wearable, it accepts the connection to begin fostering. Fostering occurs when the base station signals the wearable that the environment is "unsafe". After sending this message, the base station does not accept any data communication from the wearable other than what is shown in Figure 7 in the confirmation (reply) packet to maintain security within the base station. If the wearable attempts to send unexpected information, our app assumes it is an attacker attempting to gain access to the base station, which alerts its user regarding the unsafe environment, and quiets its personal fog.

Once the wearable sends its confirmation packet, it disconnects and quiets itself, making it invisible to devices that have not already attempted a connection to it. It stores all sensor information it collects to be broadcast to its base station when it is properly reconnected. When quieted, our app on the wearable refuses all connection attempts that are not from its original base station, as described in Section 4.3.

Fostering does not occur if the base station informs the wearable that the environment is "safe". With this information, the wearable does not initially assume that it is truly in a secure environment because the base station could be used by an attacker. In this case, the wearable does not provide any personal information to the potential attacker, as the attacker does not know the service ID used by the fostered devices personal fog. Instead, the wearable attempts to connect to other valid base stations. If no base station identified in the first inquiry informs the wearable that it is unsafe, the wearable remains active and accepts communication from devices which attempt connection using the service ID unique to its personal fog, which we describe in Section 5.3.

To validate this solution, we set up a personal fog containing device B (Figure 6) as a wearable device and device A as the base station for one user. Device C represents another user's wearable which lost contact with its base station. Once device C recognizes its disconnection, it attempts find a base station to foster with. When a base station is found, it attempts to connect. Upon connection, A informs C that the state of the current environment is unsafe. C stops all transmission and remains in a state awaiting reconnection. A second experiment has A inform C that the state of the current environment is safe. Since there are no other base stations available, C does not send information to A and remains actively collecting data until it can transmit it to its base station.

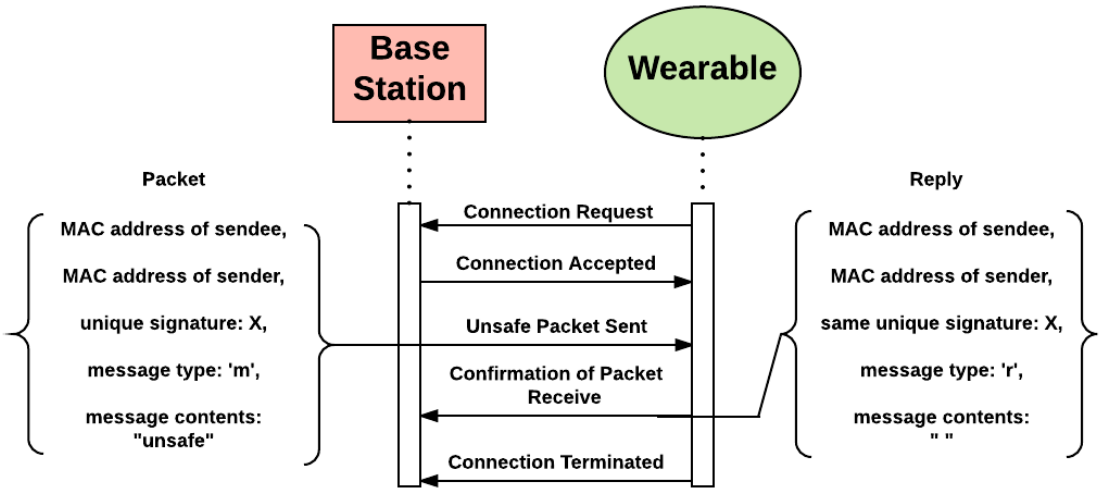

Figure 7. Wearable fostering protocol when environment is unsafe

5.2. Fostering a base station

In some instances, wearables may remain connected to their base station, but the base station might not be informed that their environment is "unsafe." This might be due to a missed update from the cloud or an attacker changing environments.

To better secure personal devices and utilize local expertise, base stations from separate fog networks can connect to each other, as a form of fostering, to inform one another of perceived changes in the environment.

For example, if a security conscious user, located in a coffee shop, noticed a potential attacker or threat, a user employing our app will designate himself as unsafe, thus shutting off the communication of personal data within his or her personal fog network. However, other personal fog networks in the area managed by less aware users, which run the app, might not be immediately privy to the potential threat. Fostering introduces the awareness to share this knowledge through the temporary connection of base stations in distinct personal fogs.

Once a base station has switched to "unsafe" mode and communicated the state to its personal fog, it runs an inquiry to discover other base stations in the area. Responding base stations treat this unsafe base station as a wearable and accept a connection, expecting to foster the device. Our unsafe base station sends "unsafe" to the fostering base station, as shown in Figure 7. Because the external base station is not expecting information in a confirmation packet it assumes it is unsafe and informs the user, currently with a pop-up notification, that it may be in an insecure environment. The user is then given the choice to ignore the situation and continue sending and receiving personal information from their personal fog or to heed the warning and quiet its personal fog devices.

To validate this solution, we shifted the device C (Figure 6) into a base station configuration and connected a Bluetooth speaker playing music to it. We then set the primary fog devices, A and B, into an insecure environment. The primary fog immediately attempted to connect to device C and sent an unsafe message. Upon receipt, we heeded the warning and stopped communication, causing our Bluetooth speaker to stop receiving music while remaining connected.

5.3. Intra-fog synchronization

Fostering wearables and fostering base stations provide two scenarios where devices from two independent fog networks might establish inter-fog communications to share security information about the environment. Once inter-fog fostering has occurred for a wearable and it has quieted itself due to an "unsafe" message, intra-fog fostering can be used to propagate its knowledge to the wearables within its original personal fog. Intra-fog synchronization, or wearable to wearable fostering, is needed if the base station of personal fog in question has not received the unsafe information or a wearable receives an "unsafe" message through fostering after being disconnected from the base station.

For this type of fostering, the previously fostered wearable with the "unsafe" knowledge performs an

inquiry specifically looking for wearables with the private service ID known only to members of its personal fog. It connects to all wearables that respond and sends them the "unsafe" message. This protocol is different from what is shown in Figure 7 because the wearable sending the inquiry is also the wearable sending the "unsafe" message. In this case, the receiving wearable reacts as if it is receiving the message from a fostering base station. Thus, it quiets itself and disconnects.

To maliciously use the intra-fog communication method, an attacker would need to know the private fog service ID. As this ID is not shared publicly, it is unlikely they will be able to access it. If the attacker somehow manages to gain access to this private ID, they are only able to quiet the wearables of the personal fog, preventing any personal information from being obtained by the attacker.

To validate intra-fog communication on the architecture, we set our three Raspberry Pis (Figure 6) up as one fog with two wearables, B and C, with A, as a base station connected to a Bluetooth speaker. Once device A became insecure, it connected to one of the wearable devices, in this case device B, and told them it was insecure. Device B immediately became insecure and attempted to connect to device C which was in its fog. After connecting on the service unique to its fog, it told the device C it was unsafe and device C stopped all communication. After changing the service on device A so it was seen as a member of the fog belonging to devices B and C, we told them they were secure again and all devices began functioning normally.

5.4. Addressing confidentiality, integrity, and availability

By design, our fostering app maintains our original apps focus on confidentiality. We maintain the integrity of the wearable date using our fostering approach by only acting on unsafe messages and disconnecting from a fostering device after only one message. Our app remains available to those around it through the base stations at all times, but will not allow an attacker to gain any information because of the instant disconnection. In addition, data and knowledge of unsafe environments increases availability than in the prior solution.

6. Evaluation

While we can demonstrate a working prototype, we need to show that a real-world application is possible moving forward. To do this, we need to show that, for a given area, we can ensure that all or most devices running our application can foster their wearables, if needed, with at least one other device. To test how this would work in a real-world environment, we created a simulation which allowed users with Bluetooth wearables to move around a pre-set environment. We allowed modification of the Bluetooth communication range, the number of users of our app, the size of the area the users were in, and the speed at which users could move.

We tested this method on three different sized locations, 1000×1000 ft., 500×500 ft., and 250×250 ft., with 10, 30, 50, and 100 users in the area with the app. We define a time-step as every 2 seconds and assume that each user can move a maximum of 25 ft. per time step, which puts us at the maximum speed of 8.5 miles per hour. This speed is about average running pace of an adult. This speed does not flood the network with data, a concern in a crowded environment sending information to the cloud via Wi-Fi, but is also fast enough that we are able to ensure that we do not miss any connections that may be made between moving devices. We also assume a Bluetooth range of 50 ft., which is well within consistent Bluetooth communication range, which can range from less than 33 ft. all the way up to 328 ft., with most devices hitting between 33 and 60 ft. [3]. Each simulation was run for a total of 40 seconds, or 20 time-steps. We ran each setting 100 times to average out any inconsistencies that may occur in individual runs.

The results of this simulation can be seen in Table 1. As we should expect, the more crowded a general area is, the more often all nodes are able to connect to at least one other node within 40 seconds. This connection shares only one thing, if the fog is in a secure environment or not. No other personal information is shared between devices or fogs.

It is important to note that each of the sizes we tested is larger than an average coffee shop. This shows that, even for a relatively low number of users, our application is viable for almost any traditional retail location. In fact, our smallest tested size, 250×250 ft., is more square footage than an acre of land. Even with this huge size, we still only had 4.7% of devices fail to connect to another device over 100 runs of our simulation with as few as 10 users.

Table 1. Percentage of nodes which failed to connect to at least one other node

% of Nodes Never Connected		Number of Users			
		10	30	50	100
Size of Location	250	4.7	0.333333	0.04	0
	500	32.7	5.5	1.5	0.18
	1000	72.5	38.03333	21	5.3

7. Conclusion and future work

In this paper, we extended our previous work regarding Bluetooth privacy and security in insecure environments by introducing the concept of a personal fog to respond to potential security or privacy threats more quickly. Additionally, we introduced the concept of fostering fogs to allow for expert opinions of insecure environments, allowing co-located users of our application to respond to newly developing threats more quickly. Finally, we showed our application is feasible on wearable devices which have control of Bluetooth communication by testing our application on a testbed built using Raspberry Pi 3s, and showed the feasibility of our app in the wild at various user densities.

This app still has room for improvement. Primarily, it is still possible for an attacker to prevent a large group of users from communicating with their wearable devices by using our app and claiming they are insecure. While this is not a problem from a data interception standpoint, it could cause users to stop trusting our app if they are always being told they are unsafe. Additionally, our app does require an existing user base with at least some users being security conscious enough to recognize unsafe environments where an attacker may be eavesdropping. Without a somewhat large initial security conscious user base, it is possible for an attacker to flood the system with "safe" signals at a given location and ensure our app would never recognize the insecurity of that location.

Moving forward, we plan to continue working with the concept of a personal fog with wearable devices, their base station, and the cloud to provide increased data security and privacy in insecure environments. We plan to examine our fog system with wearable devices which connect to additional sensors. This behavior is already being seen in consumer devices, such as the Apple Watch and AirPods.

There is also a need for greater analysis of our fostering method to ensure that no additional security threats are introduced, including an attacker being able to lower battery life of wearable devices through attacking a device with this app running on it. As there is currently not a formal definition of trust in relation to Bluetooth device communication, this research would greatly benefit from a study examining this. As Bluetooth is always improving, this research will need to be updated with newer versions of the Bluetooth standard to ensure that no new security holes are created. This includes looking into security issues arising from Bluetooth 5G. Finally, there is a need to examine possible security attacks on wearable devices more deeply and how our method can be used to provide additional security and privacy to a user.

8. References

[1] Abie, H. and Balasingham, I., "Risk-Based Adaptive Security for Smart IoT in eHealth," Proceedings of the 7th International Conference on Body Area Networks, 2012.

[2] "crackle, crack Bluetooth Smart (BLE) encryption" Accessed 2017. http://lacklustre.net/projects/crackle/

[3] "Dispelling Common Bluetooth Misconceptions" Accessed 2017. https://www.sans.edu/cyber-research/security-laboratory/article/bluetooth

[4] Giang, N.K., Leung, V.C.M, and Lea, R., "On Developing Smart Transportation Applications in Fog Computing Paradigm", ACM DIVANet, Malta, 2016.

[5] Hong, Y., Lillethun, D., Ramachandran, U., Ottenwälder, B., and Koldehofe, B., "Mobile Fog: A Programming Model for Large-Scale Applications on the Internet of Things," Proceedings of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing. 2013.

[6] Kattepur, A., Dohare, H., Mushunuri, V., Rath, H.K., and Simha, A., "Resource Constrained Offloading in Fog Computing," Proceedings of the 1st Workshop on Middleware for Edge Clouds & Cloudlets - MECC 16, 2016.

[7] NIST, "Guide to Bluetooth Security Revision 2" 2017.

[8] "Project Ubertooth," Accessed 2017. http://ubertooth.sourceforge.net/

[9] Vaquero, L.M. and Rodero-Merino, L., "Finding your Way in the Fog," ACM SIGCOMM Computer Communication Review, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[10] Walter, C., Hale, M.L., and Gamble, R.F. "Imposing Security Awareness on Wearables", Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems, p. 29-35, 2016.

[11] Walter, C., Riley, I., He, X., Robards, E., and Gamble, R.F., "Toward Predicting Secure Environments for Wearable Devices," Proceedings of the 50th Hawaii International Conference on System Sciences, 2017.

[12] Yi, S., Li, C., and Li, Q., "A Survey of Fog Computing," Proceedings of the 2015 Workshop on Mobile Big Data - Mobidata 15, 2015.