Adrasteia: A Smartphone App for Securing Legacy Mobile Medical Devices

Vahab Pournaghshband
Computer Science Department
California State University, Northridge
vahab@csun.edu

David Meyer
Computer Science Department
University of California, Los Angeles
meyerdave4@ucla.edu

Michael Holyland
IEEE Member
michael.y.holyland@ieee.org

Majid Sarrafzadeh
Computer Science Department
University of California, Los Angeles
majid@cs.ucla.edu

Peter Reiher
Computer Science Department
University of California, Los Angeles
reiher@cs.ucla.edu

***Abstract*—For a variety of reasons, Bluetooth-enabled mobile medical devices are often not designed with security in mind. As a result, many of them are open to malicious attacks, notably man-in-the-middle (MITM) attacks. For some classes of mobile medical devices, such as pulse oximeters, a MITM attack may have little impact on the safety and privacy of the patient. For more essential devices, such as pacemakers and insulin pumps, protection against MITM attacks can have fatal consequences. Though future medical devices may be designed more securely, a significant portion of existing medical devices still use an overly-trusting procedure to communicate with their desired access point. Thus, these legacy devices are still at risk of attack. This paper presents the design and implementation of an Android application to act as a personal security device (PSD) that defends against MITM attacks. To the best of our knowledge, this is the first smartphone application designed for this purpose. The use of this PSD requires no changes to either the medical device or its monitoring software, offers protection for millions of existing devices, and adds an insignificant amount of overhead to the original functionality of the medical device. Furthermore, the PSD is easily obtainable by anyone with an Android smartphone. We evaluate our defense approach by analyzing its robustness against various attacks, and we conclude with a discussion of future applications of our defense mechanism.**

***Index Terms*—Medical device security, Man-in-the-middle attack**

I. Introduction

According to studies, over 500 million people will be using health applications by the year 2015 [1]. In 2005 there were approximately 245,000 insulin pump users, and that number is expected to grow at a rate of 9% from 2009 to 2016 [2]. There are 25 million people with wireless implantable medical devices (IMD) in the U.S. alone, according to Hanna et al [8], and about 300,000 new IMDs are implanted every year.

The U.S. Food and Drug Administration (FDA) approved a Bluetooth-enabled medical device for the first time in 2003 [12]. Dozens of devices have been introduced to the U.S. market since then, with functions ranging from life-sustaining to life-supporting. Many manufacturers have not addressed the security risks of current mobile medical systems, despite the widely recognized need for increased security [3], [7], [9], [11], and have provided little security for either the devices themselves, or for the data they create and transmit.

One critical aspect of device protection is communications security. Typically, a mobile medical device communicates with a healthcare facility by asking an intermediate computer to forward the device's signals. Since such devices can usually be used with little or zero configuration by either the patient or the provider, there is no shortage of opportunity for attackers to trick the device into communicating with an attacker's machine instead of the intended intermediate computer. Whether authentic or malicious, this communication between the medical device and the intermediary is wireless, making it especially susceptible to eavesdropping and injections.

Attacks on medical devices can have severe consequences. Attackers can potentially cause the devices to operate in a dangerous or lethal manner. Consider, for example, a pulse monitoring system that uses Bluetooth to communicate with a patient's laptop, which in turn forwards heart rate data to the patient's doctor in real time. If an attack can alter the data to fake a heart attack, the doctor may call for unnecessary emergency measures, taking away medical resources from someone who may actually need them at that time. This alteration of data seems innocuous compared to the reverse scenario—the attacker can conceal the actual signs of a heart attack, and the patient will not receive the attention that they immediately need. This is just one example of the ways a medical device can cause harm when its security has been compromised. Researchers have demonstrated that insulin delivery systems and implantable cardioverter defibrillators can also be force to operate in a life-threatening manner to the patient [6].

In our prior work [14], we introduced the concept of the personal security device. In this paper we present an Android app implementation of the PSD, and discuss the advantages and disadvantages of this approach. We chose to defend the Nonin Onyx II 9550 fingertip pulse oximeter, a typical Bluetooth mobile medical device introduced to the U.S. market

in 2008. It monitors pulse rate and blood oxygen saturation levels for as long as the patient is using the device, and passes this data on to an access point (AP) at a range of several meters, over a Bluetooth connection. Although we conducted this experiment with a pulse oximeter, our defense can be applied to other devices, such as the A&D Medical UA-767PBT blood pressure monitor, with little modification. Fig. 1 depicts the general system design.

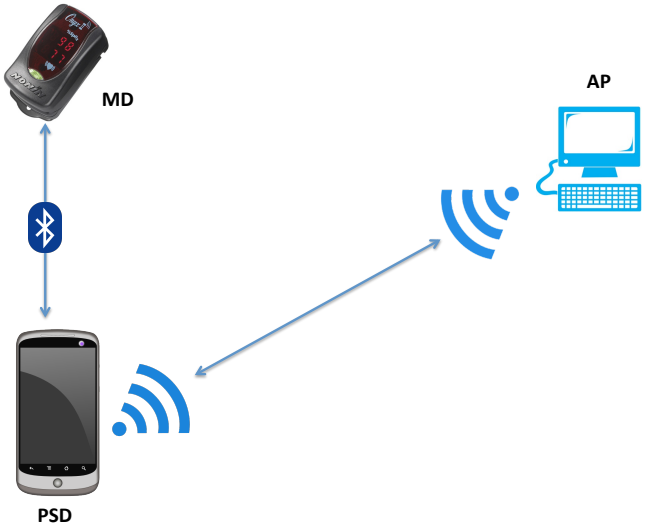

Fig. 1. How data is communicated between the medical device, personal security device, and access point.

Our defense approach does not require us to rebuild or alter the legacy devices in any way. With our Android application running, the patient's smartphone becomes a personal security device (PSD). To keep the medical device's resources available for purely medical functions, the PSD offloads security duties from the medical device. These duties include encrypting communications between the medical device and the AP, authenticating critical device commands coming in from the AP, and monitoring location to determine likelihood of attack.

The organization of this paper is as follows: Section II presents related work, followed by sections discussing the design and implementation of the PSD in sections III and IV, respectively. We discuss the threat model and robustness analyze in section V, and the limitations and challenges in section VI. Section VII takes the PSD's usability into consideration. To conclude the paper, future work and conclusions are presented in sections VIII and IX.

II. Related Work

In recent years, the security of mobile medical devices has received special attention for the generally acknowledged problem that it presents [3], [7], [9], [11]. In the wake of this increased attention, there has been some work on demonstrating attacks against various mobile medical devices [6], [10]. Other work has focused on implementing or recommending defensive approaches against these kinds of attacks [5], [6], [10], [15], [17]. Among those proposed defensive approaches, Amulet [16], Shield [5], and IMDGuard [17], all require a special-purpose third-party device to facilitate security. Also, Denning et al. [4] propose a class of devices called communication cloakers that would share secret keys with an IMD (implantable medical device) and act as a third-party mediator in the IMD's communications with external programmers. The communication cloaker preserves IMD battery power by bearing the burden of the verification of incoming requests, and can be recharged easily. Though it ignores all incoming communications from unauthorized programmers, the medical staff can remove the cloaker in emergency situations in order to access the IMD.

Rather than enabling compatibility with legacy devices, IMDGuard proposes changes in the design of future IMDs for a more secure system. By design, Amulet does not work with existing devices since it requires changes to the existing mHealth system—it asks the medical sensor to verify that it is indeed the right Amulet before connecting. Shield is the only solution that is designed to work with existing and even already implanted IMDs by requiring no changes to the device. Shield receives and jams an IMD's messages at the same time to prevent other listeners from decoding those messages, protecting the IMD so that only the authorized intermediary is able to decode them. Shield also protects the patient by jamming unauthorized commands. However, the idea behind Shield may not be practical for many mobile medical devices that operate on widely-used radio technologies such as Bluetooth or 802.11, because of the potential legal ramifications of jamming those signals and the nature of the radio technologies themselves.

The most recently proposed relevant project is an external monitor called MedMon [18], an external monitor which snoops on all radio frequency (RF) wireless communications to and from IWMDs (implantable or wearable medical devices). It uses anomaly detection to identify potentially malicious exchanges. When a potentially malicious transaction is detected, MedMon is capable of reacting either passively (user notification) or actively (jamming the packets). MedMon protects the body-area network against integrity attacks by acting like a firewall, and against battery-draining attacks, but it does not protect patient confidentiality and privacy. The most attractive benefit of MedMon is its compatibility with existing IWMDs—it is not necessary to modify any hardware or software to defend the patient.

III. Overall Design

The PSD is designed to act as a secure medium that facilitates communication between the MD and the AP. Since the PSD will need to connect to both the MD and the AP for the communication to take place, it does not matter whether the patient chooses to first connect the PSD to the MD or to the AP.

The patient can start the medical data securing process by connecting the PSD Android application with the medical device, which in this case is the Onyx II 9550 fingertip

pulse oximeter, via a Bluetooth connection. In the case of the Onyx II 9550 fingertip pulse oximeter, this is an RFCOMM connection, but we would use whichever Bluetooth profile the MD would use.

Then, the patient continues by securing the PSD to AP connection. This is done by first selecting the WiFi network the AP is in, connecting to it, and then connecting to the specific AP machine and port over an AES secure TCP connection. For this, the patient only needs to know SSID of the WiFi network. Lastly, the PSD, using the pre-shared 128-bit AES key, can authenticate itself to the AP and encrypt all packets sent across that link. Fig. 2 illustrates this process.

Lastly, the patient simply provides the PSD with the pre-shared 128-bit AES key, so that the PSD can authenticate itself to the AP and encrypt all packets sent across that link (Fig. 2).

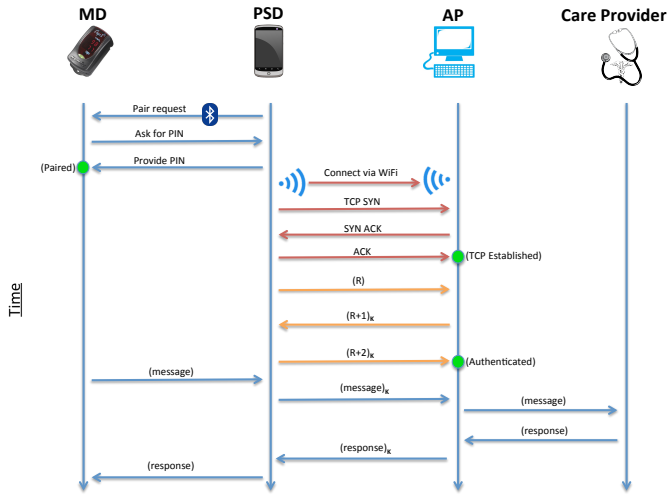

Fig. 2. Secure transfer of data from patient to provider and vice versa. R is nonce, and K is the pre-shared 128-bit AES key.

Location Data

Most patients will use their devices from locations that they trust, such as their homes or their doctors' offices. If the patient's location is known to the PSD, this information can be employed to further protect the patient from attack. Our implementation of the PSD uses GPS coordinates to enhance patient security in two ways.

First, many medical devices are capable of performing actions that directly impact the patient's immediate health, such as releasing insulin or administering an electrical impulse to the heart. The PSD keeps a best estimate of patient location, and thus always knows whether the patient is in a trusted location or an untrusted one. If a critical command arrives at the PSD, the PSD will not ask the medical device to execute that command unless it came from a trusted location.

Secondly, constantly updating location coordinates assists in the detection of suspicious activity. If the Bluetooth connection between the MD and the PSD keeps dropping in the patient's home (trusted location), it may be that the MD is running low on battery. But if the Bluetooth connection keeps dropping in a coffee shop (untrusted), the likelihood that an attacker is deliberately jamming the connection has increased. If this is the case, the PSD will issue an alarm, alerting the user of a potential attack and suggesting courses of action to defend against it, such as shutting off the medical device or moving to a trusted location. These are just two applications of using location data to make the PSD smarter about security.

IV. Implementation

The PSD App is composed of a background Service that manages the Bluetooth connection from the MD and the WiFi connection to the AP, and an Activity that lets the patient manage the Service.

A. Activity

The Activity allows the patient to monitor the status of the connection and see if any significant event has occurred. The following figure shows the home screen user interface (Fig. 3):

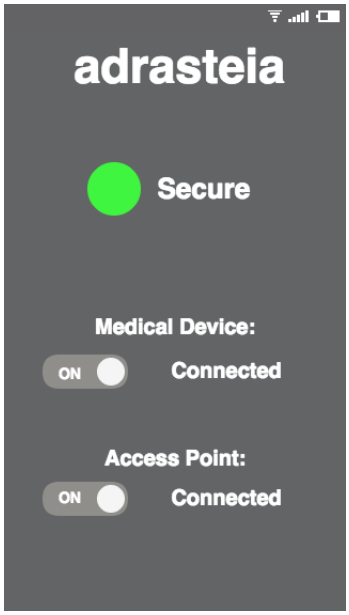

Fig. 3. All essential application information is displayed on the home screen with a user-friendly interface.

- The patient scans for nearby Bluetooth devices by touching the appropriate UI switch, and chooses the correct device from a menu of available devices.
- The patient selects the AP by touching the appropriate switch, first by selecting the WiFi network from a drop-down list of a WiFi scan.
- Once both connections, to the AP and MD, are made, the main Activity launches a Service and passes an Intent to the Service with the desired connection end points.
- Lastly, the main Activity polls the Service for updates when the patient wants to reconfigure the connection.

Alerts of significant events will pop-up on the notification bar, but the activity is where the patient can go to retrieve more information about those alerts. The notification bar will display a medical cross icon when the application status is normal (Fig. 4), and a medical cross icon with a warning exclamation mark to accompany an alert to a significant event (Fig. 5).

Fig. 4. Normal

Fig. 5. Alert!

B. Service

The Service maintains the connections to both the medical device over Bluetooth, and the AP over WiFi, as well as alerts the patient to security threats.

- The Service copies data from the MD to the AP, and vice versa, by reading from one InputStream buffer and writing to the other's OutputStream buffer. Since currently there is no way of doing non-blocking Bluetooth connections on Android, two separate threads are launched to handle the Bluetooth to WiFi, and vice versa, data transfer; one for incoming, and one for outgoing data.
- If any of the connections time-out, or if the sockets are simply closed or broken, the main thread is notified which subsequently alerts the patient.
- The main Service thread is also responsible for listening to alerts of non-secure GPS location, as well as all other threat notifications.
- If a threat is detected the Service will try to rectify the situation, which, for example, in the case of a lost WiFi connection, will be to retry the connection. If the situation cannot be resolved, the patient is notified of the potential threat.
- Lastly, the Service is polled by the Activity for more information about the threat, or for general connection information.

V. Threat Model and Robustness Analysis

There are many ways for an attacker to compromise the security of the medical device. In this section, we will enumerate the possible attacks that can be leveraged against mobile medical devices, and explain how our PSD defends against them.

A. Link from PSD to AP

Since we have complete control over this link, we can make the connection arbitrarily secure by using strong authentication and encryption protocols, limited only by resource availability. This link is susceptible to Bluetooth or 802.11 jamming as a denial of service attack. While the PSD cannot defend against this attack, it can easily detect the attack alert the patient.

B. Link from PSD to MD

If the PSD unexpectedly finds itself no longer connected to the medical device. One of three things may have happened: the medical device has run out of battery, the patient has voluntarily turned off the device, or an attacker has dropped the Bluetooth connection so as to immediately connect himself with the medical device before the PSD can recover. The first two scenarios are benign, but in the third scenario, the medical device becomes undiscoverable and the PSD will not be able to pair with it again. There are a number of ways for an attacker to exploit exclusive connection with the medical device:

1) Reading patient's real-time medical data in that short window of time: This is a violation of patient privacy at best, and a health risk at worst.

2) Ask the MD for stored sensitive data: Perhaps more dangerous than exposing real-time medical data is exposing stored data, which may include personal information and the patient's medical history.

3) Sending unauthorized commands to the MD: Perhaps more dangerous than exposing real-time medical data is exposing stored data, which may include personal information and the patient's medical history.

4) Changing device settings: The attacker can benefit from changing the settings on the medical device, perhaps to make future attacks on that device easier.

Because of the danger that the third scenario poses to the patient, our PSD treats all three lost connections equally by notifying the patient every time the connection between the MD and the PSD has dropped. If the patient was expecting to receive a notification (having just turned off the medical device), they can ignore the alert. If the patient was not expecting to be notified of a dropped connection, they can follow the alert's suggested courses of action for minimizing the damage done in case an attack did occur. Depending on the nature of the device, this may be as simple as turning the device off.

VI. Limitations and Challenges

In this section, we present some fundamental limitations and challenges associated with implementing a personal security device on the Android platform.

1) Being that the PSD is an Android application, it suffers from having to contest for resources with other applications running on the device. For our purposes, this could mean having to delay the transmission of medical data while our Service is not scheduled on the CPU. Similarly, we could have delayed data caused by the Bluetooth or WiFi stack having non-PSD packets getting processed instead of the PSD's. This may be partially

rectifiable, by having our packets have higher priority over non-essential applications. Similarly, we could set the PSD application to not be targeted by the Android OS's memory garbage collection, by running the Service as a foreground service.

2) A typical smartphone is limited to the radio technologies of GPS, Bluetooth, and WiFi. However, some medical devices use other radio technologies, such as Medical Implant Communication Service (MICS). The PSD's security features are thus not extendable to medical devices that use these technologies.

3) Since there is only a single WiFi adapter, and a single Bluetooth adapter on most smart phones, we would be limited in the number of MDs and APs. Because we have to choose which WiFi network to connect to, and the WiFi card can only access one at a time, we can only connect to APs that are on the same network as . However, we could initiate and maintain multiple TCP connections which multiple APs on the same network. As for the max amount of Bluetooth MDs that we could connect to at a time.

4) Conducting a MITM attack between the medical device and the access point is not possible, because our security design removes all direct communication between the medical device and the access point. However, it is still possible, though markedly more difficult, for an attacker to insert himself in between the medical device and the PSD. To defend against this attack, we would need only to make one reasonable assumption the medical device and PSD are geographically very close to each other (less than two feet of separation).
   With this assumption, our defense approach is to exploit that close proximity to ensure that both the MD and PSD are really talking to each other. Ideally, we would simply weaken the broadcasting power of the medical devices Bluetooth signal so that other devices can only discover the medical device if they are extremely close to it. However, it is not possible to alter the signal strength of the medical device's Bluetooth module.
   The next best approach is to reduce the Bluetooth signal strength of the PSD. Unfortunately, it is not possible to alter the signal strength of the PSD. The only way for the PSD to imitate a signal weakening effect is to measure the received signal strength indication (RSSI) of the Bluetooth connection that the MD transmits to the PSD, calculate how far away the MD must be by using the inverse square law, and then pair with the MD only if it is determined to be within two feet of the PSD. However, RSSI measurements are notoriously unreliable, since they are dependent on external factors such as battery life, interference, diffraction, and absorption. Since the measurements fluctuate so wildly, this security feature would be impractical to implement. As such, we did not implement this feature.

5) As with any new code, there is a potential for security exploits. It is possible, for example, that by using this application the MD to AP, the patient would be worse off due to someone hacking the Android device through the web. In this scenario the patient could also be lulled into a false sense of security that may not have existed without a "security app."

VII. Discussion of Usability

People of all ages use mobile medical devices to monitor and maintain their health, but since health declines with age, it is understandable that the majority of mobile medical device users are elderly. As is the case when introducing any technology to a less-experienced audience, the highest design priority of this application is usability. With an intuitively designed launch screen and interface, the patient should immediately know where they are in the application's life-cycle and what they can do from there. The graphical user interface, hereafter referred to as the GUI, consists of a primary screen from which the user can access all of the application's features. To make the GUI easier to navigate, the font size is enlarged and the colors are chosen to contrast with each other so that the screen can be seen in even the brightest daylight. From the primary screen, the patient can see the status of their smartphone's connection to both the medical device and the access point, and manually assign those connections. The status code is color coded like a stoplight so as to maintain familiarity and comfort – green for secure (no suspicious activity), yellow for secure connect in an untrusted location, and red for suspicious activity.

First time use will trigger a message that asks the user to press the Settings button to connect to a medical device. The button is placed in the settings tab so that the user is less-likely to accidentally click on it and restart the setup wizard. This button ushers the patient into the setup wizard, where they choose how they want to connect to the medical device (either Bluetooth or WiFi). If the patient doesnt know which way to connect, they will be able to download a device package update from either Google Play or the medical devices website while they wait at the doctor's office (doctor can help the patient if necessary), that will automatically configure the patient's medical device connections, and save those connections for later use.

After the setup wizard successfully exits, the patient will see that the two switches on the bottom of the screen are in the ON position, indicating that the respective connections are active. The user will be able to toggle these switches ON and OFF, where an ON toggle will ask the user to manually configure the respective connection. With both connections active, the color-coded status will be set appropriately to match the threat-level of the local environment. Otherwise, the color-coded status will disappear from view to indicate that the security feature is inactive.

The alarm is tripped for a combination of reasons. If the user is in an untrusted location as determined by GPS, the application will be more vigilant in its monitoring of dropped Bluetooth connections. If the connection is dropped multiple

times within a certain time frame, in an untrusted location, the application will cause the smartphone to vibrate and make a noise (perhaps it can even ring like a phone call, more user-friendly because some elderly people are hard of hearing). When the patient looks at the phone, the default alarm message will notify the user of a potential attack, and ask them to take action. When the medical device is known, the default alarm message can be augmented with suggested actions, best displayed as an animation. For example, for the pulse oximeter, the alarm message can ask the patient to turn off the pulse oximeter by displaying an animation of a finger being removed from a clamp. For other types of devices, a more appropriate course of action will be suggested.

VIII. Future Work

Thorough and extensive usability studies need to be conducted, such as: how much time it takes for patients of different ages to properly use this mobile application. This study must include people to be chosen with different technical background, as this application should be easy for someone with little technical background to operate. Perhaps more importantly, this study should target the elderly, as they are likely candidates for this application. Further, usability studies must include those with limited mobility or arthritis, as these conditions would likely be prevalent in those needing assistive medical devices.

The PSD should not be limited to an Android device, and we should examine the possibility of a new PSD developed into a self-contained, specialized device. Having it as a special-purpose piece of hardware has some theoretical advantages over the smart-phone solution, specifically the downsides of Android that were addressed in the limitation and challenges section. We believe a dedicated embedded solution, such as the Arduino board, is a suitable choice. This implementation and a thorough comparison analysis between these two approaches is left to future work.

It is important to implement an energy-aware application because if the phone fails due to insufficient power, the security of the medical device will be compromised. That, in itself, could be a major point of vulnerability. We will analyze and improve power performance in future work using fine-grained energy performance analysis profilers, such as Eprof [13] and PowerTutor app, which measure split-time and utilization-based energy consumption.

IX. Conclusions

Bluetooth-enabled mobile medical devices still pose a great risk to their users due to their use of an overly trusting procedure to communicate with their desired data dropoff AP. This risk can be moderate, as is the case of an attacker stealing data from a pulse oximeter, to severe, as is the case of an attacker sending malicious commands to a pacemaker. This paper detailed a possible solution to the MITM attacks that legacy Bluetooth-enabled MDs are susceptible to. We implemented a phone application to implement a personal security device on the Android platform. We discussed the implementation and various scenarios to show how this app can protect against many possible attacks while we laid out fundamental limitations which exists in the smart phone solution.

Acknowledgments

This work is supported by the National Science Foundation, CNS-1116371.

References

[1] "Mobile Health Apps: What Do You Use?" http://www.cbc.ca/news/pointofview/2010/12/mobile-health-apps-what-do-you-use.html.

[2] "Insulin pumps - global pipeline analysis, opportunity assessment and market forecasts to 2016, globaldata," *Global Data*, 2010.

[3] S. Avancha, A. Baxi, and D. Kotz, "Privacy in mobile technology for personal healthcare," *ACM Computing Surveys*, 2012.

[4] T. Denning, K. Fu, and T. Kohno, "Absence makes the heart grow fonder: New directions for implantable medical device security," in *Proceedings of the 3rd Conference on Hot topics in security*. USENIX Association, 2008, p. 5.

[5] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu, "They can hear your heartbeats: non-invasive security for implantable medical devices," in *Proc. of the ACM SIGCOMM Conference*, New York, NY, USA, 2011, pp. 2–13. [Online]. Available: http://doi.acm.org/10.1145/2018436.2018438

[6] D. Halperin, T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *Security and Privacy. IEEE Symposium on*. IEEE, 2008, pp. 129–142.

[7] D. Halperin, T. Kohno, T. Heydt-Benjamin, K. Fu, and W. Maisel, "Security and privacy for implantable medical devices," *Pervasive Computing*, 2008.

[8] K. Hanna, *Innovation and invention in medical devices: workshop summary*. National Academies Press, 2001.

[9] D. Kotz, "A threat taxonomy for mHealth privacy," in *Communication Systems and Networks (COMSNETS), 3rd International Conference on*. IEEE, 2011, pp. 1–6.

[10] C. Li, A. Raghunathan, and N. Jha, "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system," in *e-Health Networking Applications and Services, 13th IEEE International Conference on*, 2011, pp. 150–156.

[11] W. Maisel, "Safety issues involving medical devices," *JAMA: the journal of the American Medical Association*, vol. 294, no. 8, pp. 955–958, 2005.

[12] L. Ott, "The evolution of Bluetooth in wireless medical devices," *Socket Mobile, Inc. White Papers*, 2010.

[13] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof," in *Proceedings of the 7th ACM european conference on Computer Systems*. ACM, 2012, pp. 29–42.

[14] V. Pournaghshband, M. Sarrafzadeh, and P. L. Reiher, "Securing legacy mobile medical devices," in *3rd International Conference on Wireless Mobile Communication and Healthcare*, ser. MobiHealth12. Springer, 2012.

[15] K. Rasmussen, C. Castelluccia, T. Heydt-Benjamin, and S. Capkun, "Proximity-based access control for implantable medical devices," in *Proc. of 16th ACM Conference on Computer and Communications security*, 2009.

[16] J. Sorber, M. Shin, R. Peterson, C. Cornelius, S. Mare, A. Prasad, Z. Marois, E. Smithayer, and D. Kotz, "An amulet for trustworthy wearable mHealth," in *HotMobile*. New York, NY, USA: ACM, 2012, pp. 7:1–7:6. [Online]. Available: http://doi.acm.org/10.1145/2162081.2162092

[17] F. Xu, Z. Qin, C. Tan, B. Wang, and Q. Li, "IMDguard: Securing implantable medical devices with the external wearable guardian," in *INFOCOM*, 2011.

[18] M. Zhang, A. Raghunathan, and N. K. Jha, "Medmon: Securing medical devices through wireless monitoring and anomaly detection," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 7, no. 6, pp. 871–881, 2013.