Security Vulnerabilities For IMD:Pacemakers

Christopher Williams
ECpE
Iowa State University
Ames IA State USA
chrisjw1@iastate.edu

Shanell Hurst
ECpE
Iowa State University
Ames IA State USA
sahurst@iastate.edu

Rishab Kinnerkar
ECpE
Iowa State University
Ames IA State USA
riskin@iastate.edu

ABSTRACT

Implanted medical devices are moving from latent technologies to more modernized ones, allowing consumers not to be tethered to a medical facility. While these changes were designed with patient and healthcare providers in mind, it has created a new market for security features in these devices. The purpose of our project is to review these new requirements and create a secure bluetooth protocol for accessing implanted medical devices. We propose tackling these new challenges using a two factor hardware authenticator, and have simulated a variety of attack scenarios on this system.

The purpose of our project is to review these new requirements and create a secure bluetooth protocol for accessing implanted medical devices. Because of varying technologies in different devices, we chose to hone in on pacemakers which operate using bluetooth low energy protocol (2.4-2.485GHz) and are resistant to noise. Latent technological pacemakers work in 400 MHz bands. Even though patients currently have this embedded technology, it is outdated and no longer implemented in modern devices. Thus, focusing on future implementations is necessary to begin to combat targeted attacks on victims.

Our secure system architecture is designed as an extra layer of security on top of what Bluetooth LE already provides to hopefully deter attackers from targeting IoT devices. The security protocol would be implemented on an external device where authentication would occur. Our project is focused on using a two factor hardware based authenticator using both Bluetooth LE and an RSA signature. Our purpose is to make these devices more difficult to hack with the intent to deter attackers from targeting these devices. We were able to successfully send and receive messages, encrypt the data being sent and create a message signature using RSA. The code we implemented should be able to ward off possible attacks against the system.

KEYWORDS

Implanted Medical Devices, Wireless, Bluetooth

INTRODUCTION

Implanted medical devices have critical roles ranging from delivering medication, monitoring body functions along with providing support for or replacing organs all together. Although these implants can be life saving,undergoing the procedure of implementation comes with great risk of infection, rejection, or device failure[5]. Thus, it is best if they are replaced as infrequently as possible.

A major threat on the rise for patients and medical care providers is the risk of being hacked. Hacking can leak information, deny services, and even compromise control of IMDs in serious cases. Some attacks of concern include: ransomware, man-in-the-middle (packet injection and eavesdropping), replay, brute force on encryption and denial of service attacks.

Wireless vulnerabilities can become a gateway into hospital networking systems allowing hackers to access a range of medical equipment and patient records. Access to these critical devices open extortion pathways for malicious attacks to all patients undergoing treatment in medical facilities by holding any IoT device hostage. This in turn traps the medical facilities and patients under attack [6].

New technologies have allowed various implanted medical devices to get more sophisticated with the use of bluetooth communication for data transfer or for device-configuration. These devices are mainly designed to operate efficiently for the tasks they are configured to take on and therefore have a lot of hardware constraints on them. However, both modern and latent technologies have some constraints such as low memory, processing and limited battery life. These design implementations make implanted medical devices susceptible to attacks and limits security implementations that can be placed on the devices.

In many cases, these embedded devices are vital to the patient's health and could become dangerous if compromised. The purpose of a pacemaker is to deliver controlled electrical stimulus to the heart, specific to each individual patients needs, in order to maintain an effective heart rhythm by resynchronizing the heart throughout the duration of the patient's treatment period. The treatment period can be temporary or permanent. Although there are numerous types of pacemakers from leadless to rate responsive, all pacemaker surgery is considered high risk. As recently as August 2018 the FDA issued a voluntary recall of 465,000 Abbott pacemakers citing security concerns causing a lot of inconvenience for the patients. These are some instances Wireless security solutions need to be implemented as a step towards a safer product for customers [2,5,7].

Focusing on Medtronic's Azure pacemaker model, communication from the pacemaker occurs directly to some bluetooth low energy enabled mobile platform via BlueSync™ Technology. The pacemaker requires the mobile device to have the MyCareLink Heart™ mobile application to communicate with the healthcare provider and perform patient data collection and monitoring. The data is encrypted inside the pacemaker using *NIST: National Institute of Standards and Technology before it is sent to the mobile device. The application then sends the data to a Medtronic programmer which is in the medical office.

The Azure model only accepts programming from Medtronic programmers that are located in close proximity. This model is not connected to the Internet and does not have an IP address. However, as far as we can see with our research thus far it is not required for the physicians to authenticate to a programmer, and the programmer devices themselves don't authenticate to implantable pacemaker devices. According to researchers Billy Rios and Jonathan Butts of WhiteScope IO, "Any pacemaker programmer can reprogram any pacemaker from the same manufacturer," [4,10]. Although this is a software concern at the programmer and pacemaker, which is not the focus of this project, we can still add a layer of security by incorporating two factor authentication on an external device for all wireless communications.

There has been a lot of research done in this field and over the years researchers have come up with different methods for increasing the security; however, more security measures can be taken to further enhance patient safety. Our project is a proposed system for further securing these new devices which communicate via Bluetooth. This system involves hardware based, two factor authentication for authorizing operations on the device. We will demonstrate this system's robustness against common cyber attacks and show how the usage of our device makes interacting with IMDs easier and safer.

Literature review

In 2011, a research group at MIT worked on a radio-based device called IMD Shield which the patient wears around their neck [1]. MIT addressed several attack issues of concern on implanted medical devices such as confidentiality and unauthorized commands by creating an external device called a *shield*. The benefits of the shield is that no modifications are made to an already implanted medical device. The shield acts as a jammer-cum-receiver by using radio frequencies to jam incoming messages while blocking unauthorized commands. The *shield* will simultaneously send out packets at the same time it is jamming [1].

The shield is great for implanted medical devices with embedded latent technologies; however, it will not work for modernized Bluetooth technologies. Innovations in pacemaker devices have incorporated Bluetooth Low Energy (LE) technologies which allow users to have multi connections with various electronic devices, multi broadcasting features such as point-to-point and broadcasting while utilizing little battery and improving reliable operations. The low energy frequency band operates at 2.4 GHz and will frequency-hop over a 40 channel spectrum for data transmission.

The shield functions by broadcasting noise on 400MHz band. Researchers have demonstrated how easy it is to hack into pacemakers which use low radio frequencies. Their settings can easily be altered and attacks can also render the pacemakers dysfunctional. With readily available tools like an oscilloscope and software radios it is possible to hack into pacemakers running on simple low radio frequency protocols [1]. However it is not so easy to do so on bluetooth.

Bluetooth is designed to be resistant to noise so multiple people can use the same frequencies. Because Bluetooth operates over a 40 channel spectrum, creating noise on all channels would be very difficult and likely unwanted. Also, Bluetooth allows for multiple security options up to government grade security [3].

Hardware based cryptography can now be used for secure communications at a low power cost [11]. In the early 2000's engineers avoided implementing a sophisticated cryptographic system on the pacemaker due to limitations on the pacemaker in terms of memory, energy and inalterability [1]. In 2003, a group of researchers from Arizona State University had worked on securing IMD's using biosensors wherein the patients biometrics would be externally used as authentication keys. Using biometrics for authentication is not a trivial problem but these researchers were able to come up with a theoretical system call BioSec in which they showed that it is possible to achieve security with low energy consumption [11]. This was never practically implemented because of the safety concerns such as being unable to temporarily adjust the pacemaker in emergency situations [1].

Pacemakers that don't use HTTPS in their communication are hackable in research environments. This was demonstrated in 2018 at Black hat and Defcon events by a group of researchers [10]. The problem with medtronic pacemakers is that the data sent to the programmer from the pacemaker is not delivered via an encrypted HTTPS connection [10]. Thus the data is not digitally signed and if it were to come from a malicious source the pacemaker nor the programmer would be able to possibly recognize this which makes it possible for hackers to potentially deliver or deny shocks to the patient.

In cryptography, One Time Pad (OTP) is an encryption technique that cannot be cracked but is difficult due to the continuous changing of keys for every encryption and the high computation and memory requirements. In 2015, researchers from Macquarie University were able to come up with a security protocol which was based on a variant of the One Time Pad in which the patient's Electrocardiogram(ECG) data was being used for keys [9]. This security system provided protection from eavesdropper and active adversaries as was demonstrated by these researchers [9]. Although, they demonstrated the strength of their security system and the theoretical way of obtaining the keys yet this research lacked the practical implementation aspect. So having a secure system would be of no good if it is implemented along with a poor architecture.

Methodology

Our system is secured by using encrypted channels and a hardware two factor authenticator. The premise of the system is that accessing the device should only be possible if the hardware authenticator associated with that pacemaker pushes the authentication button after the change request is submitted. With proper encryption, this prevents a wide range of possible attacks on the device. Chiefly, this prevents against eavesdropping, man in the middle attacks, replay attacks, and (potentially) denial of service attacks.

Our authenticator is a fairly simple device which contains a 256 bit private key. It servers the function of responding to "challenges" from the pacemaker which verify that the authenticator is present and that the patient is in the office with the healthcare professional. When it receives a message from the pacemaker it will prompt the user to push and hold the authenticate button to allow the change to be made. The reason for this is to require physical action on the part of the patient in order to make a change. This should protect the patient in scenarios where an adversary has connected to their pacemaker outside of a healthcare setting. In the next section we go over the actual protocol for using this authenticator in a healthcare office.

Encryption in our system is done using the RSA public key algorithm. This scheme creates two keys for each device, a public key used for encrypting incoming messages and a private key used for decrypting those messages. Additionally, the private key is used to sign all sent messages to verify the identity of the device sending the message. The main advantage of this scheme is that the private key is never transmitted outside each device, so impersonating them is very difficult relative to a symmetric encryption authentication. The downside of this algorithm is that it uses more energy than many symmetric key algorithms and it requires more bytes to transmit a certain amount of plaintext data.

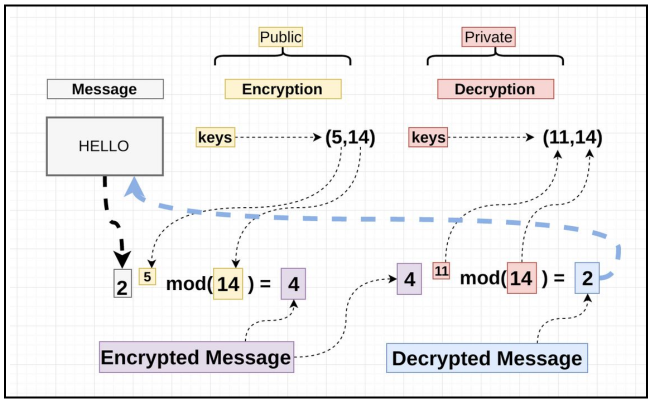

Figure 1: RSA Encryption Diagram [8]

Submitting Changes to the Device

1. The patient is in the healthcare office with their pacemaker connected to a programmer. The patient has their authenticator with them. The programmer is able to speak to the pacemaker via Bluetooth Low Energy and can also send messages to the backend.

2. The healthcare provider enters a change to be made to the pacemaker. The programmer signs this message with its private key and sends an encrypted message to the pacemaker. The message is encrypted with the pacemaker's public key.

3. The pacemaker sends a challenge request to the authenticator. This message simply requests that the authenticator respond with a valid message for the current time. This message is signed with the pacemaker private key and encrypted with the authenticator public key.

4. The authenticator prompts the user for input. If the user pushes and holds the button, then the authenticator will respond to the challenge with a signed and encrypted message for the current time.

5. If the pacemaker gets the response to the authentication challenge and it is verified as being valid, then it will act on the request from the programmer. Alternatively, if a request doesn't come within a short time threshold, then it will consider the change unwanted. It will respond to the pacemaker with the

result of the operation, and keep an internal record of the event as well.

6. The programmer will forward this encrypted response to the backend so that it can be logged appropriately.

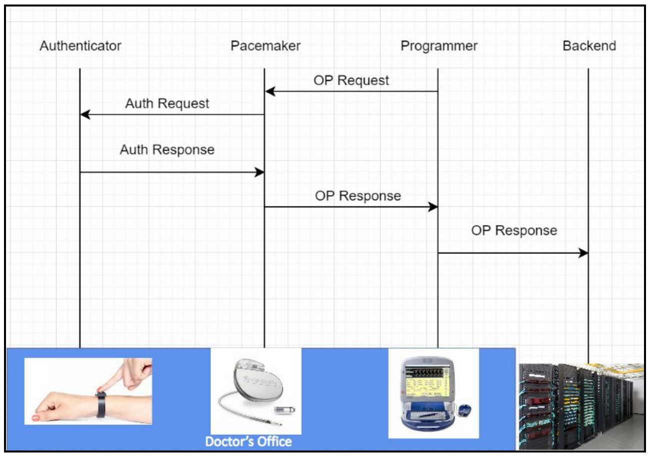

Figure 2: Protocol Diagram

Project Implementation

When we started working on our project, we originally chose to use Omnet++ as an event simulator to demonstrate our protocol. We were able to successfully set up our network simulation for the project. However, when we started to work on the implementation of the cryptography libraries in C++ we found it to be very difficult to incorporate with the timeframe we had left. After researching various simulators, we decided to use an alternative event simulator called Simpy. Simpy would grant us access to simpler cryptography libraries we felt could be successfully implemented in the time remaining for the project.

Simpy is a general purpose discrete event simulation library written in Python 3. It models machines as "processes", which are Python functions that run in a loop. These functions yield "events", which cause the process to wait a virtual amount of time. Because Simpy is general purpose, we had to create the code to do basic networking logic. All of this code is included in a class called "network_module", which is meant to mimic the functionality of the cSimpleModule class in Omnet++. This class can "send" messages to the other machines, created encrypted message bytes, and create an RSA signature as well as verify signatures from the messages.

The class "programmer" methods are able to Initiate request for change to pacemaker, connect the pacemaker to internet and forward responses from operation to server.

Another class called the "pacemaker" was implemented with the purpose of receiving messages from programmer and authenticator. This class will requests authentication from authenticator after the operator makes a request. For the purposes of this project we always assume the button is pushed to initiate the authenticator to start the authorization process. This code also is programmed to give a response back to programmer.

Class "authenticator" responds to authentication challenge from the pacemaker. A challenge is the request for authentication where the timestamp is different every time. Simply signs a valid message for the current time Timestamp to ensures messages are only valid for a short time. So although we did not test any replay attacks, this code is written to mitigate this issue.

The class "backend" logs the results of pacemaker operations. This class could also be used to serve additional purposes in more advanced version of protocol. For example, if this project was implemented it would be necessary to incorporate code for emergency access that would not require the use of the authenticator.

Project Results

We had our simulation output a simple log file that specifies which events occurred at which time. As stated before we only created a basic test scenario which involved a single operation request which is authenticated by the authenticator. All of our messages were properly signed and decrypted by the other devices. If we had time, we would have created additional test cases which involved corrupted messages, replayed messages, etc. Although we didn't directly handle these in test cases, our code still takes these possibilities into account. Additionally, due to the way we implemented our code in Simpy there wasn't always a delay when sending messages between devices. This didn't change the protocol itself, but could be improved if another simulation of our protocol was created.

Milestones

In this section we discuss the key milestones we were able to achieve for our project. In the first phase we set up a project that everybody could download and build. This was done by March 4. In the second phase we were able to create the basic implementation of all the components using Omnet++ which was done by March 29. In the third phase we re-created the implementation of all the basic components using simpy, implemented the encryption and communication protocol and simulated basic test cases which was done by April 29.

Limitations

1. In our implementation we had fixed message delivery rates among the different network components and the

effect of potential message delays was not implemented by us.

2. The packets which the network components exchanged had only the timestamps signed by the sender for authentication purposes. This would prevent replay attacks but in the real version we would have signed the entire body of the message for more security.

3. Our implementation was one meant to be run on any wireless protocol and we did not specifically test its performance on bluetooth low energy protocol.

4. Our implementation didn't have an emergency backdoor situation implemented. We had discussed implementing this, but didn't create one for lack of time.

Conclusion

Implanted medical device technologies have evolved in recent years to include both bluetooth low energy and data encryption for communication. While these designs were implemented in part to provide convenience for both the medical professionals and patients, security vulnerabilities have become a frightening reality.

Although more latent technologies are difficult to address with infrastructure designs, we feel our project will build on past research performed and add an additional layer of safety to modernized pacemaker devices. Using a hardware based two factor authentication device with pacemakers adds another layer of security. This requires physical action on the part of the patient in order for modifications to be made to the pacemaker. This can mitigate possible wireless attacks trying to reprogram pacemakers outside of the healthcare office.

Our project will work with the current device technologies of Bluetooth low energy frequencies. Although we were not able to test various attacks, we hope this project will inspire the incorporation of the additional authentication and safety measures on these devices to deter attackers from targeting them.

WORKS CITED

[1] S. Gollakota, H. Hassanie, B. Ransford , and D. Katabi. They Can Hear Your Heartbeats: Non-Invasive Security Implantable Medical Devices, August 15-19, 2011, Toronto, Ontario, Canada.
[2] S.Ibrahimi. A Secure Communication Model For The Pacemaker: A balance between security mechanisms and emergency access. August 2014.
[3] https://www.bluetooth.com/bluetooth-technology/radio-versions, Feb 14, 2019.
[4]Amanda Pedersen,2017 How Medtronic's Pacemakers Are Now Harder to Hack,https://www.mddionline.com/how-medtronics-pacemakers-are-now-harder-hack.
[5] Author unknown, FDA Issues A warning about Hackable Implanted Cardiac Devices last updated August 28 2018, written Jan 12, 2017 https://www.fda.gov/MedicalDevices/ProductsandMedicalProcedures/ImplantsandProsthetics/FDA.
[6] Lily Hay Newman, Medical Devices Are the Next Security Nightmare. https://www.wired.com/2017/03/medical-devices-next-security-nightmare/3/2/17.
[7] https://www.hcahamilton.com/implantable-cardioverter-defibrillator.
[8] Figure 1: https://hackernoon.com/how-does-rsa-work-f44918df914b
[9] Zheng, Guanglou, et al. "Encryption for implantable medical devices using modified one-time pads." *IEEE Access* 3 (2015): 825-836.
[10] Ms. Smith, 2018 "Hacking Pacemakers, Insulin Pumps and Patients' Vital Signs in Real Time."
https://www.csoonline.com/article/3296633/security/hacking-pacemakers-insulin-pumps-and-patients-vital-signs-in-real-time.html.
[11]Cherukuri,K.K.Venkatasubramanian,andS.K.S.Gupta.Biosec:A biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. In International Conference on Parallel Processing Workshops.