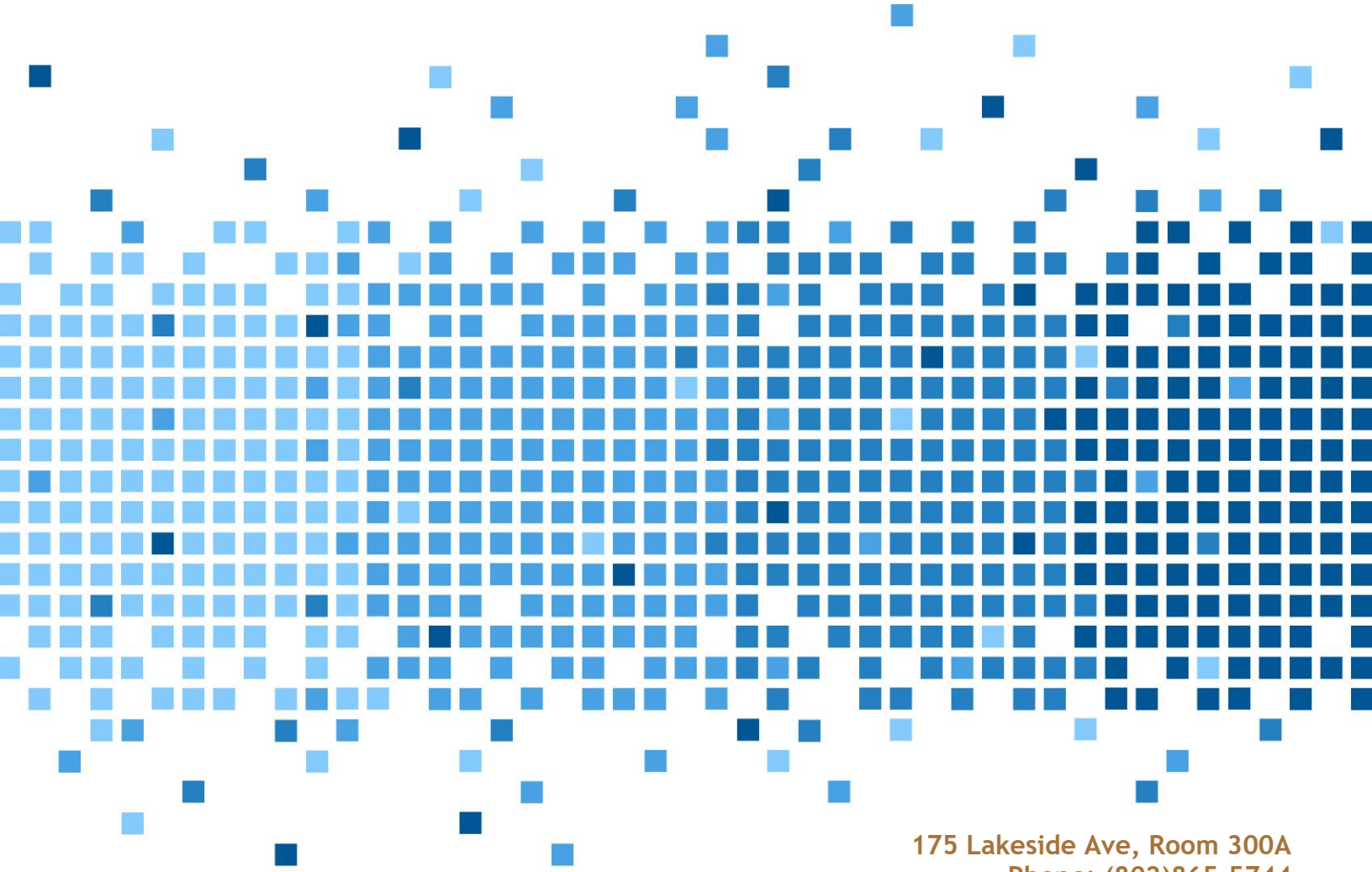
CHAMPLAIN COLLEGE | LCDI Leahy Center for Digital Investigation

[Bluetooth Security]



175 Lakeside Ave, Room 300A
Phone: (802)865-5744
Fax: (802)865-6446
http://www.lcdi.champlain.edu

02/02/2016

Disclaimer:

This document contains information based on research that has been gathered by employee(s) of The Senator Patrick Leahy Center for Digital Investigation (LCDI). The data contained in this project is submitted voluntarily and is unaudited. Every effort has been made by LCDI to assure the accuracy and reliability of the data contained in this report. However, LCDI nor any of our employees make no representation, warranty or guarantee in connection with this report and hereby expressly disclaims any liability or responsibility for loss or damage resulting from use of this data. Information in this report can be downloaded and redistributed by any person or persons. Any redistribution must maintain the LCDI logo and any references from this report must be properly annotated.

Contents

Introduction

Technology is advancing exponentially, and as it does, our desire and need to protect digitally stored personal information becomes tantamount to the continued adoption of these newer technologies. However, it stands to reason that, occasionally, we miss certain possible vectors for security flaws on the basis that the technology stack underlying these systems is overly complex and/or assumed to "just work." Bluetooth stands as one of these "untouched" technologies that, while well into adoption and widespread use, has not been exhaustively analyzed for possible vulnerabilities, even though independent sources suggest there may be a great multitude. Over the course of this project, the LCDI has been taking steps to critically examine Bluetooth for any flaws that might threaten the security of data that passes over Bluetooth connections.

Background:

The Bluetooth project was created at the start of the 2016 spring semester in response to LCDI employees expressing interest in a focused examination of the capabilities and vulnerabilities of the Bluetooth protocol during the 2015 fall semester.

Purpose and Scope:

The purpose of this research is to provide evidence that Bluetooth is not impenetrable, bringing awareness to the presence of critical vulnerabilities in both hardware and software stacks of Bluetooth-enabled devices. The hope is that consumers, business owners, and manufacturers alike will draw important conclusions regarding their role in understanding and securing Bluetooth-enabled devices from information capture/sniffing.

Research Questions:

1. What vulnerabilities currently exist in most Bluetooth-enabled devices?
2. How are these vulnerabilities exploited?
3. How does one stop these vulnerabilities from being exploited or go about patching these vulnerabilities?

Terminology:

Bluebugger – Command line tool used to gain access to a Bluetooth device without authentication.

Bluecat – Command line tool used to debug Bluetooth devices.

Bluesnarf – Command line tool used to gain access to a Bluetooth device without authentication in order to steal information.

Crackle – A Linux tool developed to exploit a flaw in the Bluetooth low-energy pairing process that allows the user to decrypt captured data packets.

Kali – A Linux distribution dedicated to penetration testing.

Kismet – Packet-sniffing tool normally used to view Wi-Fi packets that also has functionality to be used alongside the Ubertooth to sniff Bluetooth packets.

Ubertooth One – An open-source Bluetooth monitoring and development platform that allows a user to monitor and capture data sent over the low-energy frequencies that Bluetooth uses to connect devices.

Wireshark – An application used to analyze network traffic.

Methodology and Methods

For this project, we'll primarily be using the Ubertooth One Bluetooth sniffer, a small USB device that allows for the entire 2.4 GHZ band (encompassing WiFi and Bluetooth) to be sniffed for Bluetooth and Bluetooth Low-Energy (BTLE) packets. We'll be utilizing Kismet, a piece of packet-sniffing software, with Ubertooth plugins that allow us to sniff Bluetooth traffic (rather than the Wi-Fi traffic that Kismet usually interprets). In addition, we will be utilizing WireShark, another packet-sniffing application, for sniffing and interpreting BTLE packets. We'll be storing most of the data in pcapng files, which will be analyzed for sensitive data or ways to possibly manipulate the devices that transmitted the packets. Bluetooth packets are divided into different layers, due to the different levels of software on what is known as a "Bluetooth stack," which requires interoperability between a lower-level device layer (for the Bluetooth antenna) and an upper-level software layer (for communicating packets to the host operating system). Because of these different packet layers, analysis of vulnerabilities will be categorized and discussed in terms of the layer that is at hand.

Equipment Used

Table 1: Hardware

Device	OS Version
Dell Precision M2300 (Laptop)	Kali Linux 2016.1
LCDI Project Computer (Desktop)	Kali Linux 2016.1
Ubertooth One	One
Jawbone Jambox	

Table 2: Software

Software	Version	Comments
Kali Linux	2016.1	Installed on Laptop
Crackle	0.1	Installed on Laptop
Wireshark	2.0.2	Installed on Laptop

Data Collection:

Data collection began upon receiving the Ubertooth One. Once it was set up and all pre-requisites were installed, we used our control devices to generate Bluetooth traffic. The traffic was captured using the Ubertooth and saved to a pcap file. The pcap file was then run through Crackle to decrypt the data and passed to Wireshark where we were able to examine the newly decrypted data.

Approved Bluetooth-enabled devices were also tested for common exploits and vulnerabilities in different situations and connection configurations (all connected, phone to watch, tablet to headphones, headphones to phone, etc.) with different exploits and Bluetooth-cracking tools. The devices were also used to test the range of Bluetooth technology to determine the feasibility of long-range attacks.

Analysis

When we began this project, we knew there were already some vulnerabilities in Bluetooth that were public knowledge. As a result, we decided that we wanted to test out these vulnerabilities along with some that we found in our initial research. By using the Ubertooth One, we were also able to record and analyze traffic between two Bluetooth devices and discover what information can be obtained through regular network traffic. This was very important to our research since the Ubertooth One is an open source device that anyone can purchase and potentially use. We also used several devices to test the possible range of Bluetooth technology in a parking lot and public park.

Results

One of the vulnerabilities that we discovered was the fact that Bluetooth allows two devices that have been previously paired to connect without further authentication. Since Bluetooth relies on MAC addresses to identify one device from another, this presents the possibility of MAC address spoofing. A MAC address is a unique identifier that is made up of six groups of two hexadecimal digits. Every electronic device is given their own individual MAC address, usually denoting the make and model of the device with a few characters that make it unique. Unfortunately, there are programs that are publicly available that allow temporary "spoofing," or changing, of a MAC address.

As a result, anyone with malicious purposes could spoof their MAC address to that of a device that has already been paired with the target. This would allow the attacker to gain access to the target device without alerting the target of any malicious activity. In addition, many devices are now being implemented with a feature called Smart Lock that automatically unlocks a smartphone if it is in the presence of another Bluetooth device that it has been paired with. This feature, though it has a valid purpose, only emphasizes the need for a fix to this vulnerability.

On the bright side, it appears that developers have fixed some vulnerabilities that had been discovered in the past. While testing programs like Bluesnarf and Bluebugger, both of which are hacking tools designed to exploit Bluetooth in order to access and steal information, it was discovered that when these programs try to connect to a Bluetooth device, they are still required to authenticate themselves. This allows the user of the target device to deny access to the attacker, thus, defeating the hacking program.

When testing the range of Bluetooth, we looked at several scenarios like a Jawbone speaker connected to an iPhone and an iPhone connected to a Computer. The test on the Jawbone speaker connected to an iPhone was conducted in a parking lot, while the rest of the tests were conducted in the Burlington City waterfront park. Table 1 below shows the results of the test conducted in the parking lot.

Table 1: Jawbone Speaker to iPhone

Jawbone Speaker to iPhone				Average (ft.)
Initial Packet Loss (ft.)	42	47	50	**46.33**
Fatal Packet Loss (ft.)	86	94	90	**90**
Reconnect (ft.)	58	53	57	**56**

In this test, we used an iPhone 5C to connect and play music through a Jawbone Jambox speaker. We then walked away from the speaker with the iPhone until the music started to become choppy. We recorded this distance as the initial packet loss. We then kept walking until the music stopped altogether and recorded the distance as fatal packet loss. Lastly, we walked back towards the speaker until the phone was able to reconnect to the speaker. This distance was recorded as the reconnect distance. We figured that we were getting a lot of interference since we were running the tests in a parking lot full of vehicles, so we decided to move to the Burlington City waterfront park. A test run between an iPhone and a Computer was then conducted and the results are shown in Table 2.

Table 2: iPhone to Computer

iPhone to Computer				Average (ft.)
Initial Packet Loss	59	64	60	**61**
Fatal Packet Loss	88	95	92	**91.67**
Reconnect	54	50	51	**51.67**

For this second test, we connected an iPhone to a Dell Laptop and played music through the Laptop's speakers. We then proceeded to run the same tests as those conducted in Table 1. We ended up getting pretty similar results as in the first test, concluding that interference due to cars was minimal. For the third and final test, we connected a Samsung Phone to a pair of Jaybird Bluebud X headphones and played music. We ran the same tests as the last two trials with the only difference being that one of us held the phone about one foot off the ground as the person wearing the headphones walked away. The results obtained in this last test can be found in Table 3 and show that when a device is raised off the ground, its signal is greatly increased.

Table 3: Headset to Samsung Phone

Headset to Samsung Phone				Average (ft.)
Initial Packet Loss	130	139	114	**127.67**
Fatal Packet Loss	250	219	219	**229.33**
Reconnect	255	280	263	**266**

Conclusion

As Bluetooth is already implemented in most devices and becomes even more widely used, the security that comes along with it is just as important as the functionality itself. Businesses that develop the technology need to make sure that there are no security vulnerabilities that can be used to steal sensitive data or do harm to the end-users. Bluetooth technology has also been developed and slowly implemented into things such as motorized skateboards and cars. This has presented the opportunity for hackers to develop ways to exploit the technology for actual physical harm. As a result, serious inquiries into how these vulnerabilities can be fixed is strongly recommended.

Further Work

After completing our research, we have determined that Bluetooth technology, despite already being implemented in almost every aspect of our lives, still contains several security vulnerabilities that could result in sensitive information being stolen for malicious purposes. This is a huge security risk for businesses and consumers who rely on technology to secure their information. Thus, continued research into how these vulnerabilities can be fixed is strongly recommended as well as research into the security vulnerabilities contained in future versions of the Bluetooth protocol.

References

"Now I Wanna Sniff Some Bluetooth: Sniffing and Cracking Bluetooth with the UbertoothOne." RSS. Security Sleuth, 13 Sept. 2015. Web. 26 Apr. 2016. <http://www.security-sleuth.com/sleuth-blog/2015/9/6/now-i-wanna-sniff-some-bluetooth-sniffing-and-cracking-bluetooth-with-the-ubertoothone>.

"Sniffing Bluetooth Packets with Kismet and Wireshark in Ubuntu 12.04." Ceres Controls. Ceres Controls, n.d. Web. 26 Apr. 2016. <http://cerescontrols.com/tutorials-3/sniffing-bluetooth-packets-with-kismet-and-wireshark-in-ubuntu-12-04/>.

Gao, Vincent. "Proximity and RSSI | Bluetooth Technology Website." Bluetooth Blog. Bluetooth, 21 Sept. 2015. Web. 26 Apr. 2016. <http://blog.bluetooth.com/proximity-and-rssi/>.

Nasim, Robayet. "Security Threats Analysis in Bluetooth-Enabled Mobile Devices." International Journal of Network Security & Its Applications IJNSA 4.3 (2012): 41-56. Arxiv. Web. 26 Apr. 2016. <http://arxiv.org/ftp/arxiv/papers/1206/1206.1482.pdf>.

Ossmann, Michael, Dominic Spill, and Mike Ryan. "Greatscottgadgets/ubertooth." GitHub. Great Scott Gadgets, n.d. Web. 26 Apr. 2016. <https://github.com/greatscottgadgets/ubertooth/>.