Assessment of Bluetooth Vulnerabilities

By

Jonte' Logan & Shyam Pema

Submitted to
The Faculty of the School of Information Technology
In Partial Fulfillment of the Requirements for
The Degree of Bachelor of Science
In Information Technology

Jonte'Logan 4/15/16
Jonte' Logan Date

Shyam Pema 4/15/16
Shyam Pema Date

Bodgan Vykhovanyuk 4/15/16
Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2016

Table of Contents

Table of Contents i
Figures and Tables iii
Abstract iv
Introduction 1
Problem Statement 1
Overview of Bluetooth Technology 2
Design 6
Project Description 6
Project Constraints 7
User Profile 8
Planning 10
Budget – Cost to Build 10
Testing 10
Devices Tested 10
Test Report 11
Overview 11
Scope 11
Objective 12
Testing Procedures 12
Pass/Fail Scenarios 13
Risks 13
Results 14
Vulnerabilities Discovered 14
Bluetooth BLE Pairing Process Flaw 14
Bluesnarfing/Bluebugging 15
Bluesmacking 16
Bluejacking 16
Framework for Securing Bluetooth Vulnerabilities 16
Keeping trusted devices paired 16
Long range Bluetooth connectivity 17

Keeping your device updated 17
Two factor Bluetooth Authentication 18
Conclusion 18
Appendix 20
Gantt Chart – Fall Semester 20
Gantt Chart – Spring Semester 23
Technology Used 26
Toothscraper 26
Ubertooth One 27
Kali Linux 30
btscanner 31
Hcitool 31
Wireshark 32
MySQL 33
phpMyAdmin 33
Bootstrap 34
Bibliography 37

Figures and Tables

Figures

Figure 1 2
Figure 2 4
Figure 3 9
Figure 4 15
Figure 5 20
Figure 6 21
Figure 7 22
Figure 8 23
Figure 9 24
Figure 10 25
Figure 11 26
Figure 12 27
Figure 13 28
Figure 14 28
Figure 15 29
Figure 16 30
Figure 17 31
Figure 18 32
Figure 19 33
Figure 20 35

Tables

Table 1 5
Table 2. 6
Table 3 9
Table 4 10
Table 5 11

Abstract

The Bluetooth protocol allows devices to connect via short range, wireless radio communication signals, which can be hijacked, therefore opening the opportunity for confidential information to be accessed. This vulnerability poses a significant threat to companies that allow employees to bring their mobile devices into the workplace and connect them to their internal network. This also affects those who leave Bluetooth enabled on their mobile devices when out in public. Attacks such as denial of service, eavesdropping, keyboard takeover, and unauthorized access to a mobile device's contacts and messages are some of the exploits that can prove to be quite impactful depending on the compromised target. Currently there is very little literature focused on the effort to secure mobile devices from Bluetooth attacks. In order to present policies and standards on preventing these exploits from occurring, research needs to be conducted on the capabilities of Bluetooth communication. With the results taken from this project, we hope to identify methods that utilize vulnerabilities in the Bluetooth protocol in order to compromise a target device, as well as present security policies that can be employed to prevent these attacks

Introduction

Problem Statement

Users of mobile devices are vulnerable to a variety of attacks ranging from downloading malicious applications to enabling Bluetooth in a public space. Bluetooth utilizes several Application Program Interface (API) tools that many users enable when they want to listen to music or make phone calls. Some users leave their Bluetooth connection unsecured, others may secure their Bluetooth connection with a password. We want to research how Bluetooth communication can be exploited and use the results to identify and provide solutions against these types of vulnerabilities. We will be looking at Bluetooth technology using mobile devices to find vulnerabilities that can be exploited. Many users can fall victim to these types of attacks, and we want to study the various methods that are available and pose solutions to them.

By not identifying vulnerabilities in using Bluetooth, users who fall victim to these attacks are susceptible to having personnel data, company data, and other important information stolen and compromised by an attacker targeting or scanning open devices. Through our research, we will research what types of attacks can be utilized through Bluetooth connections and evaluate how extensive these attacks can become. Our research will lead to developments that could resolve future potential attacks from occurring on mobile devices. We want to further understand what patterns and trends are being used within the realm of Bluetooth technology.

Overview of Bluetooth Technology

In order to assess vulnerabilities in Bluetooth, first we needed to understand how the technology works. Bluetooth is a wireless communication protocol for connecting devices in close proximity to each other. It operates over the 2.4 – 2.485 GHz frequency band. It is intended for short range use (3-100 m) between two or more devices. Common devices containing Bluetooth technology are included in virtually every device today, including cell phones, speakers, keyboards, car entertainment systems, key fobs, printers, headsets – the list goes on. When implementing Bluetooth adapters inside of devices, it is up to the manufacturer to configure the security mode of that device. In order to establish a connection between a group of devices via Bluetooth, they must first create a network called a piconet. Piconets consist of a maximum of seven slave devices that are connected to one master device. The master device controls how often and exactly when slave devices communicate with itself, as slave devices are not allowed to talk to other slave devices in the same piconet. Figure 1 shows how slave devices transmit and accept data from only a single master device.

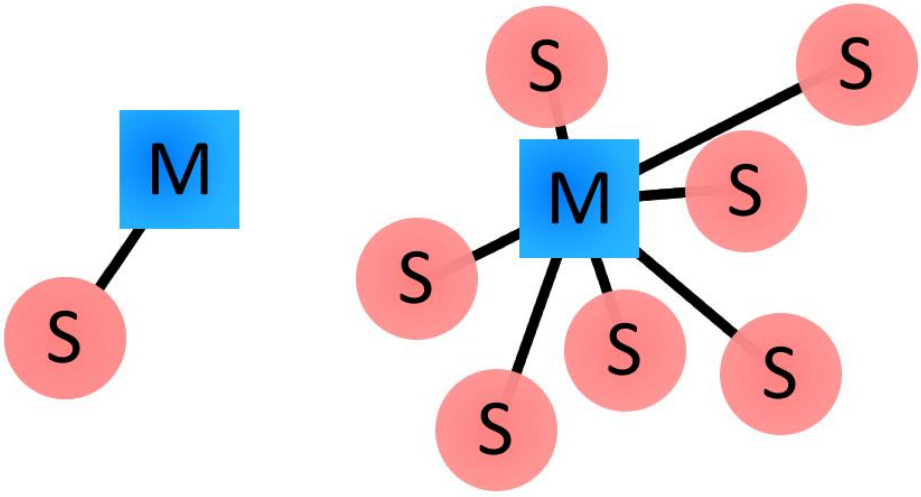


Figure 1. Illustrates a diagram of a piconet, showing how devices connect to each other in a Bluetooth network

Bluetooth vulnerabilities can arise when manufacturers do not set the correct security options that would establish secure communication between two devices. There are four security modes that increase in the capabilities for secure transmission and authentication protocols. Security Mode 1 does not utilize authentication or encryption methods when connecting to another device, making this configuration a dangerous option for devices that would be exposed to the public or that would transmit highly sensitive and confidential data. Security Mode 2 contains a manager that controls access to services that devices may request access to use. This mode begins after a physical network link connection, but before the logical link connection has a chance to completely finish. The security manager has defined policies that it enforces, based on the security configuration that an application sets. These services require authentication and authorization from the requesting device. Security Mode 3 enforces security link features to be completed before the physical network link is established. All devices participating in this security mode must use encryption and authentication when communicating. A security manager is not implemented in this mode once a device has been authenticated. The strongest security configuration is Mode 4, which includes a security manager that does not implement policies until after both the physical network link and the logical network link connections have been established. This mode is similar to the characteristics of Security Mode 3, however Mode 4 uses Secure Simple Pairing (SSP) to encrypt the keys used to authenticate devices. However, it is up to the configuration of a service on whether it will utilize encryption of authentication keys. Figure 2 shows a short description on each security mode.

Security Mode	Description
1	No security. Device operates in "promiscuous" mode allowing any other BT device to connect to it.
2	Service level security. Security measures are initiated AFTER the channel is established. Supports authentication, encryption, and authorization.
3	Link level security. Security measures are initiated BEFORE the channel is established. Supports authentication and encryption.

Figure 2. Shows the three security configurations for Bluetooth devices

Two levels a device can obtain are trusted and untrusted. Once authenticated, Bluetooth allows trusted devices access to the certain services, while blocking access to others if requested from an untrusted device. Four security levels are used within Security Mode 4, when a service is invoked. Service Levels 0 & 1 offer little protection against protecting data in transmission and at rest, as they do not require man-in-the-middle (MITM) protection, encryption of authentication keys, or involve user interaction. Security Level 2 requires implementation of encrypted keys, but not MITM protection, while Security Level 3 requires both MITM protection and the use of encrypted keys. Bluetooth devices also can be in two states that determine what types of packets they will respond or accept – discoverable and connectable. Devices in discoverable mode respond to inquiry scans, which a device usually responds to with its Bluetooth hardware device address (BD_ADDR), device name, local clock, and other attributes. Devices in connectable mode monitor a frequency for page requests, which announce that that device is ready to accept a network connection.

The more power that a device can consume will allow the range of Bluetooth to stretch across a greater distance, due to the presence of a more powerful transmitter, which often requires heavier power consumption. Table 1 shows the different classes of Bluetooth devices and their approximate range.

Class	Power	Range
Class 1	100 mW	100 meters
Class 2	2.5 mW	10 meters
Class 3	1 mW	1 meter

Table 1. Displays the power and range levels for three classes of Bluetooth transmitters.

The Bluetooth Special Interest Group (SIG) is the group responsible for developing and maintaining the technology. The group hopes to continually expand the use of its technology through research by keeping up with advancements in technology. Table 2 below shows the evolution of Bluetooth technology since its inception in 2001 with Bluetooth 1.0.

Bluetooth Version	Major Enhancements
Bluetooth 1.0	Manufacturers encountered many bugs; included mandatory Bluetooth hardware device address in transmission
Bluetooth 1.1	Fixed many of the bugs from previous version; added non-encrypted channels and Received Signal Strength Indicator (RSSI);
Bluetooth 1.2	Added Adaptive Frequency-Hopping Spread Spectrum (AFH); Flow control and retransmission nodes for L2PCAP
Bluetooth 2.0	Enhanced Data Rate (EDR) for faster transmission rates
Bluetooth 2.1	Secure Simple Pairing introduced
Bluetooth 3.0	Increased theoretical transfer speeds to 24 Mbit/s, supports Bluetooth over WLAN connections

Bluetooth 4.0	Now called “Bluetooth Smart” or “Bluetooth Low Energy (BLE)”, it offers high speed transfers using a low amount of energy
Bluetooth 4.1	Minor upgrade, enhanced connection oriented protocols and topologies were updated
Bluetooth 4.2	Link layer privacy, LE secure connections

Table 2. Displays the major enhancements of every major release of Bluetooth.

Design

Project Description

Our project consists of researching attacks that utilize Bluetooth communication. We explored attacks such as bluesnarfing, bluebugging, and bluejacking; in addition, an exploit in Bluetooth version 4.0 was also researched and described. Those methods were investigated to see what attacks are ultimately successful, the extent of damage that can be done and how we can develop a form of defense against these attacks. For the first part of our project, we used an Ubertooth One, Kali Linux, and Bluetooth capable devices to capture packet streams between the two devices, which would be viewable in Wireshark.

The second part of our project involved creating an inventory of mobile devices through Bluetooth scanning. Several tools exist to gather device information from Bluetooth-enabled devices through inquiry packet requests. This information can be useful in keeping a log of devices that are in connectable mode and open to receiving pairing requests. However, there is no central location for storing this type of information. Our tool called ToothScraper, creates an asset management database for Bluetooth-enabled devices. It scans devices and stores important information such as a device's MAC address, IP address, device name, using a preconfigured

script that imports data from a text file, that was created using Bluetooth tools available on Kali Linux. Along with the inventory, we developed a Bluetooth security framework that consists of recommendations that can be used to secure Bluetooth devices from being compromised.

Project Constraints

The problems that we experienced included finding Bluetooth exploits that work with more recent smartphones. With the research we conducted, the most common Bluetooth exploits can be executed on very early versions of Android and on pre-smartphone era cell phones, because they have older versions of Bluetooth hardware and software installed on them. Another problem we ran were ethical concerns, in terms of making sure that any Bluetooth packet decryption we choose to execute, belongs to devices that are owned by us. There were no attempts to use these intercept communications between unowned devices or to carry out attacks on devices of unsuspecting device owners. We carried out these attacks on devices that we owned, located in a safe testing environment. This was especially a concern during the process where we rooted, unlocked, and flashed a custom operating system to our Nexus 7 tablet in order to install Nethunter. Without following specific instructions made for the Android device that is being rooted, the process itself presents many opportunities for the tablet to be inoperable.

When looking at different situations, utilizing simple steps such as locking your Bluetooth device with encrypted passwords is a small start to securing Bluetooth communications. Most devices that fall victim are devices that are left discoverable. Making devices non-discoverable could serve as another solution to protect against attacks. Once we have found a defense for some of these attacks, we could choose to upload updates to devices through system updates, just like patching a machine with security updates. If certain devices

have become compromised by an attack, in most cases, the victim can just walk away and their device will resume Bluetooth communication. Restarting the device will disable the effects of some of the attacks that were performed. A more drastic option would be to erase all data and settings from the device using a factory reset, which should ensure that the device has been cleaned of all exploits.

User Profile

This table explains our target audience for this project, as well as the preferred knowledge that an IT security administrator would need to have in order to implement Bluetooth restrictions.

User Profile Form
Application: We used Kali Linux as the platform in which we utilized existing Bluetooth tools to conduct our analysis of Bluetooth packets and to analyze device information.
Potential Users: Our Bluetooth security framework and the ToothScraper tool aim discover new methods of finding vulnerable Bluetooth devices in order to patch them and provide solutions. Our main focus is for our framework to be considered both the enterprise and public environments.
Software and Interface Experience: The user must be comfortable operating Kali Linux tools from the command line. There are various tools that require extensive levels of expertise, but they can be learned through different methods and guides online, as most of the tools are open source.
Experience with Similar Applications: Most Bluetooth applications that were used have been installed on Kali Linux can be ran from the command line and configured with the use of switches.
Task Experience: Ubertooth One packages must be downloaded and installed prior to using the Ubertooth device. Using an Ubertooth One to scan for discoverable Bluetooth devices requires research in order to know what commands to utilize. There are guides available on Github for the Ubertooth One in order to guide the user through initial setup. In terms of using the

ToothScraper tool, the user needs to be able to start services, run the script, and verify the results by querying the MySQL database.
Frequency of Use: When information security professionals wish to find exploits in devices when unknown devices join their network, they can use our research that describes common exploits, in order to mitigate them. Since most corporate environments allow employees to use mobile devices for personal and business use, they can use these tools in combination with our research to develop ways to defend their company from Bluetooth attacks.

Table 3. User Profile form

The following diagram shows our design principles for this project, the capabilities of Bluetooth, as well as how the technology would be applied in different environments. This should show how we plan to progress from our initial research phase, demonstrate our understanding of that research, and how we produced a working demonstration that effectively conveyed the purpose of our project.

Figure 3. Use Case Diagram

Planning

Budget – Cost to Build

The items listed below are the materials gathered for this project. Some items were not able to be incorporated into the project because of technical restraints, and are thus noted with an asterisk.

Item	Cost
Ubertooth One	$138.95
ASUS 7-Inch 32GB Black Nexus 7 tablet*	$199.00
Wireless Bluetooth Keyboard Case	$19.95
Kali Linux	Free to download
Kali Linux Nethunter	Free to download
VMware Workstation 11	Free one-year license to CECH IT Students
Mobile devices (owned by us)	Free
Total Cost	$357.90

Table 4. Budget – Cost to Build

Testing

Devices Tested

During our initial tests to ensure that ToothScraper was configured properly, we used several different devices. Table 5 shows the scanning results from ToothScraper using various mobile devices.

Device	Discoverable by ToothScraper?
Apple iPhone 6	Yes
Apple iPhone 6S	Yes
Apple iPhone 4	Yes
Apple iPad Mini 3	Yes
Samsung Galaxy Tab 2	Yes
Google Nexus 7	Yes
Wireless Bluetooth Keyboard	Yes

Table 5. Shows Toothscraper's ability to identify Bluetooth-enabled devices.

Test Report

Overview

This section will explain ways that mobile devices can be identified through the use of Bluetooth communication. Access to this information allowed us to inventory device metadata and store it within our database. The devices used for testing purposes were an Apple iPhone 4, an Apple iPhone 6, a Google Nexus 7 tablet, a wireless Bluetooth keyboard, a Bluetooth speaker, and Bluetooth earbuds.

Scope

The extent of testing will include the tool's ability to inventory device information through Bluetooth and store it within our database. We also tested that the data was successfully imported into the database by querying it.

Objective

Our objective was to confirm that data from the text file is successfully imported into our database correctly. The test confirms if the script is collecting the correct data without major flaws.

- **How did we determine what to test?**

  We wanted to test the core components of ToothScraper and relate the results back to our framework of vulnerabilities.

- **What were your test objectives?**

  Our test objectives involved running the script to invoke the scanner and generate all the information to a text file, which is then imported into the MySQL database. This information will be viewable from a web browser. We used the organization of this information to drive our framework to assess vulnerabilities per a unique device.

Testing Procedures

1. hcitool started and collected device information, which was then saved to a text file.
2. The automated script was expected to complete the following tasks:
    - Export a text file from our selected tools
    - Import that text file and store it in the MySQL database
    - Append the data to the correct table in the database
    - Ensure all fields (Bluetooth address, device name, manufacturer, and features) are in the correct column (the column fields are still being determined)
3. PHPMyAdmin was the user interface used to configure database; it was also be used to verify that the import ran successfully

Pass/Fail Scenarios

It was expected that the script will run through the aforementioned commands without producing any errors. If there were any errors, they most likely related to specific services not being started, or that the script is unable to find a text file to import.

Risks

- There may be data validation issues if data is imported to incorrect columns
- It may take multiple tries for the Bluetooth scanning tools to receive a response from discoverable devices
- The legality of storing Bluetooth addresses in plain text without permission (testing will only be conducted using our devices, if others are used, then we would notify the owners before scanning)

Results

Vulnerabilities Discovered

Bluetooth BLE Pairing Process Flaw

During the initial pairing process between two devices using Bluetooth Low Energy (BLE) technology, the initial pairing request is sent unencrypted. This means that a third party can intercept the link key by sniffing the communications, therefore opening the possibility for a rogue device to authenticate with the stolen key. Since this link key can be used to establish a connection between two devices after accepting the correct PIN code, packet injection arises as another method of compromising communications. Once both devices generate link keys, we can compare them to ensure they are the same, encrypted communications begin. However, since the link key is able to be sniffed, the encrypted information is unsafe because the attacker has access to the key that the encryption was built upon. A tool called Crackle was created to take advantage of this flaw, which according to the website Lacklustre, where the tool's description can be found, Crackle is able to "guess or very quickly brute force the TK (temporary key) used in the pairing modes supported by most devices (Just Works and 6-digit PIN). With this TK, crackle can derive all further keys used during the encrypted session that immediately follows pairing. The LTK (long-term key) is typically exchanged in this encrypted session, and it is the key used to encrypt all future communications between the master and slave" (Lacklustre 2014).

Bluesnarfing/Bluebugging

Connects the Bluetooth-enabled victim device to pair with another device without confirmation. This allows for the modification of contacts, calendars, emails, and text messages on smartphones that are running a version of Bluetooth that is susceptible to this attack. A popular tool to exploit this vulnerability is available to use on Kali Linux, called Bluesnarfer. Bluesnarfer uses a device's BD_ADDR to scan the remote device and execute phonebook modfications such as adding or deleting entries, based on the switches included in the command. Figure X shows the available options Bluesnarfer can be configured to alter.

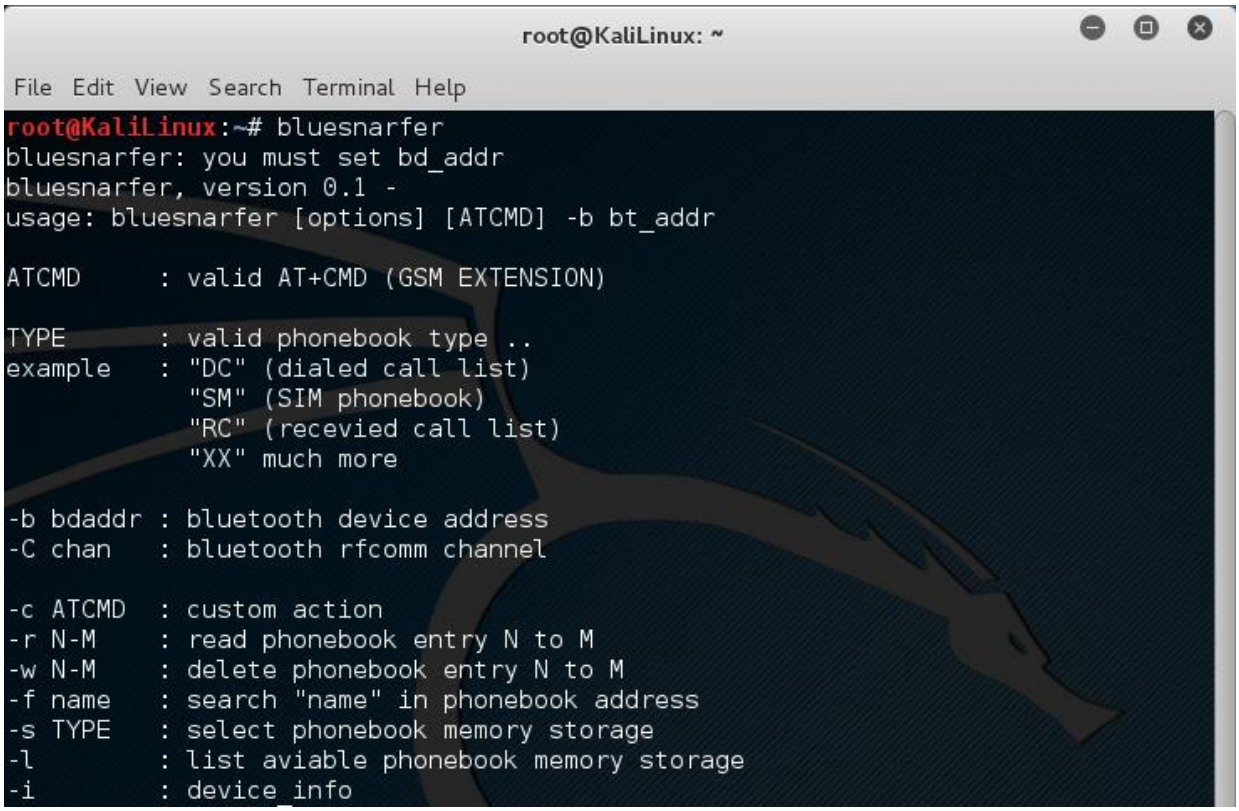


Figure 4. Displays available configuration options for Bluesnarfer

Bluesmacking

Bluesmacking attempts to overwhelm Bluetooth devices by sending a large amount of L2CAP ping packets. This can be achieved using the BlueZ stack, which provides support for Bluetooth's layers and protocols. Bluesmack is essentially a Denial of Service (DOS) attack for Bluetooth-enabled devices.

Bluejacking

Bluejacking involves sending unsolicited text messages or contact cards (otherwise known as V-Cards) using OBEX protocol. By utilizing the open discovery methods of Bluetooth communication, attackers can anonymously send contact cards containing messages to nearby Bluetooth devices without prior permission.

Framework for Securing Bluetooth Vulnerabilities

The framework below serves as a list of security recommendations for safe practices involving Bluetooth technology. These recommendations can be applied to any setting, whether they are utilized in an enterprise environment, in public locations such as coffee shops or malls, and in a home environment. It is important to implement Bluetooth security policies in order to protect the integrity of any Bluetooth-capable device.

Keeping trusted devices paired

When two devices are authenticated and paired together, they gain access full access to specific services on the other device. Assuming both devices have a legitimate need to be connected to each other, this is generally acceptable and contains low risk of malicious threats. Throughout the time these devices are paired, one of the devices could be compromised by a

malicious application seeking to exploit Bluetooth protocols. This puts the other device at risk due to their paired status, which could allow the malicious application to connect to the second device to obtain sensitive information. Our recommendation to prevent this would be to un-pair devices that have been connected longer than two months. When the devices are disconnected, each device should be scanned with anti-virus and anti-malware applications, as well as checking for any new applications that have been installed without permission.

Long range Bluetooth connectivity

Bluetooth was created with the intention of utilizing short range radio waves as its primary method of communication. A standard Class 1 Bluetooth device is capable of transmitting up to 300 feet. However, with the addition of a high gain antenna attached (through hardware modifications) onto an existing Bluetooth dongle, you can extend the range past the Class 1 specified range threshold. A Bluetooth dongle with extended range could be used to detect Bluetooth devices from a malicious actor across the street. Enabling Bluetooth in public locations can be risky, as this opens up your device to attacks such as bluejacking or bluesmacking.

Keeping your device updated

Bluetooth vulnerabilities can be found and exploited in devices that are not updated to the latest software. For example, an Airdrop exploit specific Apple's mobile devices showed that the attacker penetrated through the Bluetooth security and was able to file share taking over the device completely. This exploit could only take place if they were in range of the victim's device. Rogue devices compromised through vulnerabilities can automatically install malicious apps without being approved by the owner. The attacker could also access personal information

as well. Apple featured a patch to this vulnerability, but users must update their device to fix this patch. Most users don't update their devices regularly leaving them susceptible to their devices being compromised. Our recommendation is to keep mobile devices updated with the latest operating system release.

Two factor Bluetooth Authentication

As Bluetooth technology evolves, there should be more secure methods to secure certain Bluetooth devices other than a four pin code. Devices such as a wireless keyboard or mouse, transmit sensitive data and should use two-factor authentication. We can integrate this technology into Bluetooth's existing protocols. Protocols such as U2F (Universal Second Factor) use elliptic curve cryptography to protect the data transmitted. When pairing a device, you must put input the four pin code; but with the U2F protocol, it could generate another higher layer of encryption to protect connected devices. This prevents hackers from being able to hijack Bluetooth transmission's while in the pairing process. If two transmission sensitive Bluetooth devices are to be connected, they should use U2F security to protect their devices.

Conclusion

After completing our initial research and configuration of our demonstration, we felt confident in our ability to convey how Bluetooth works to an audience, and then demonstrate those facts into a live demo. We believe our initial demonstration successfully showed that it is possible to capture Bluetooth communication between to devices, and explained the malicious methods that can be carried out with the collection of this data using a packet capture software such as Wireshark. For the second semester, we created a tool to automatically scan for Bluetooth devices and record their metadata to a database. With the collection of this

information, we hope to simplify the process of creating a mobile device inventory by storing device data in a structurally organized manner. Our proposed security framework addresses the need of educating the public on the dangers of Bluetooth and how to protect against the aforementioned attacks. It also covers areas that we think could be improved upon in future implementations of Bluetooth. At the conclusion of our project, we showcased live Bluetooth packet capture, configured a tool that creates a mobile device inventory, and created a framework that presents security practices to protect against Bluetooth vulnerabilities.

Appendix

Gantt Chart – Fall Semester

	Task Mode	Task Name	Duration	Start	Finish
1		**Planning Phase**	**25 days**	**Mon 9/28/15**	**Mon 10/19/15**
2		Gather price quotes for equipment	7 days	Mon 9/28/15	Sun 10/4/15
3		Decide on vendor of equipment and place order	7 days	Mon 9/28/15	Sun 10/4/15
4		Research known Bluetooth vulnerabilties	7 days	Mon 9/28/15	Sun 10/4/15
5		Narrow down list of Nethunter tools to use for vulnerability testing	7 days	Mon 10/5/15	Sun 10/11/15
6		Research the capabiltiies of Nethunter	7 days	Mon 10/5/15	Sun 10/11/15
7		Become familiar with Nethunter through online research	7 days	Mon 10/5/15	Sun 10/11/15
8		Install Nethunter onto devices	7 days	Mon 10/12/15	Sun 10/18/15
9		Begin configuring equipment for baseline tests	7 days	Mon 10/12/15	Sun 10/18/15
10		Determine realistic exploits	7 days	Mon 10/12/15	Sun 10/18/15
11		**Implementation Phase**	**17 days**	**Mon 10/19/15**	**Mon 11/2/15**
12		Begin drafting final report	7 days	Mon 10/19/15	Sun 10/25/15
13		Start researching methods to	7 days	Mon 10/19/15	Sun 10/25/15

Figure 5. Gantt Chart

	Task Mode	Task Name	Duration	Start	Finish
13		Start researching methods to prevent selected Bluetooh exploits	7 days	Mon 10/19/15	Sun 10/25/15
14		Begin drafting presentation slides	7 days	Mon 10/19/15	Sun 10/25/15
15		Evaulate the effect of Bluetooth exploits	7 days	Mon 10/19/15	Sun 10/25/15
16		Decide on an exploit	7 days	Mon 10/26/15	Sun 11/1/15
17		Begin a technical/non-te write-up of the	7 days	Mon 10/26/15	Sun 11/1/15
18		Begin discussing implementation details for backup plan	7 days	Mon 10/26/15	Sun 11/1/15
19		Continue drafting presentation slides	7 days	Mon 10/26/15	Sun 11/1/15
20		**Execution Phase**	**25 days**	**Mon 11/2/15**	**Mon 11/23/15**
21		Continue drafting presentation slides	7 days	Mon 11/2/15	Sun 11/8/15
22		Update project plan, deliverables and Gantt Chart	7 days	Mon 11/2/15	Sun 11/8/15
23		Finalize presentation slides	7 days	Mon 11/9/15	Sun 11/15/15

Figure 6. Gantt Chart (cont.)

	Task Mode	Task Name	Duration	Start	Finish
24		Ensure the working demo is presentable	7 days	Mon 11/9/15	Sun 11/15/15
25		Rehearse presentation	7 days	Mon 11/9/15	Sun 11/15/15
26		**Testing Phase**	**9 days**	**Mon 11/16/15**	**Mon 11/23/15**
27		Document vulnerability details	9 days?	**Mon 11/16/15**	**Mon 11/23/15**
28		Test vulnerablity on different devices	9 days?	**Mon 11/16/15**	**Mon 11/23/15**
29		Check for consistency	9 days?	**Mon 11/16/15**	**Mon 11/23/15**

Figure 7. Gantt Chart (cont.)

Gantt Chart – Spring Semester

	Task Mode	Task Name	Duration	Start	Finish
30		**Revision Phase**	**57 days**	**Mon 1/11/16**	**Mon 2/29/16**
31		Review presentation feedback	**7 days**	**Mon 1/11/16**	**Sun 1/17/16**
32		Conduct additional research for the final paper	7 days	Mon 1/18/16	Sun 1/24/16
33		Determine new deliverables from revised scope	7 days	Mon 1/18/16	Sun 1/24/16
34		Demonstrate why the end user should be concerned	7 days	Mon 1/25/16	Sun 1/31/16
35		Explore the need for additional equipment	7 days	Mon 1/25/16	Sun 1/31/16
36		Determine need for ordering backup equipment	7 days	Mon 2/1/16	Sun 2/7/16
37		Revise working demo based off feedback	7 days	Mon 2/1/16	Sun 2/7/16
38		Continue revising final	7 days	Mon 2/8/16	Sun 2/14/16
39		Continue research	7 days	Mon 2/8/16	Sun 2/14/16
40		Review fall presentation and critique	7 days	Mon 2/15/16	Sun 2/21/16
41		Being creation of a layman audience pitch	7 days	Mon 2/15/16	Sun 2/21/16
42		Evaluate	7 days	Mon 2/15/16	Sun 2/21/16

Figure 8. Gantt Chart (cont.)

	Task Mode	Task Name	Duration	Start	Finish
41		Review fall presentation and critique	7 days	Mon 2/15/16	Sun 2/21/16
42		Being creation of a layman audience pitch	7 days	Mon 2/15/16	Sun 2/21/16
43		Evaluate creation of an enterprise focused pitch	7 days	Mon 2/15/16	Sun 2/21/16
44		Test functions of demo to ensure operations are working properly	7 days	Mon 2/22/16	Sun 2/28/16
45		Validate results and compare to what was expected	7 days	Mon 2/22/16	Sun 2/28/16
46		**Refinement Phase**	**64.63 days**	**Mon 2/29/16**	**Sun 4/24/16**
47		Begin revised demonstration walkthrough	7 days	Mon 2/29/16	Sun 3/6/16
48		Draft mockup poster and plan on a theme for the poster	7 days	Mon 2/29/16	Sun 3/6/16
49		Ensure the main parts of our research are included in the paper	7 days	Mon 3/7/16	Sun 3/13/16
50		Create presentation slides	7 days	Mon 3/7/16	Sun 3/13/16
51		Continue revising final	7 days	Mon 3/14/16	Sun 3/20/16
52		Begin	7 days	Mon 3/21/16	Sun 3/27/16

Figure 9. Gantt Chart (cont.)

	Task Mode	Task Name	Duration	Start	Finish
52		Begin brainstorming potential response to questions	7 days	Mon 3/21/16	Sun 3/27/16
53		Ensure demo and slides are coherent	7 days	Mon 3/21/16	Sun 3/27/16
54		Review fall presentation feedback	7 days	Mon 3/21/16	Sun 3/27/16
55		Prepare for presentation	7 days	Mon 3/28/16	Sun 4/3/16
56		Continue revising final report	7 days	Mon 3/28/16	Sun 4/3/16
57		Complete Final report	7 days	Mon 4/4/16	Sun 4/10/16
58		Ensure all equipment for Expo has been ordered/purchas	7 days	Mon 4/4/16	Sun 4/10/16
59		Prepare for Tech Expo	7 days	Mon 4/11/16	Sun 4/17/16
60		Gather all necessary equipment for Expo	7 days	Mon 4/11/16	Sun 4/17/16
61		Walkthrough demo and ensure presence of fallback plans	7 days	Mon 4/11/16	Sun 4/17/16
62		Edit and make final revision for submission of the final paper	7 days	Mon 4/18/16	Sun 4/24/16

Figure 10. Gantt Chart (cont.)

Technology Used

Toothscraper

ToothScraper is a tool that we created that builds a simple mobile device inventory by invoking a third-party Bluetooth scanner, called hcitool, to scan for device information and import the results into a MySQL database. Figure 10 below shows a screenshot of the script used to save and import the data.

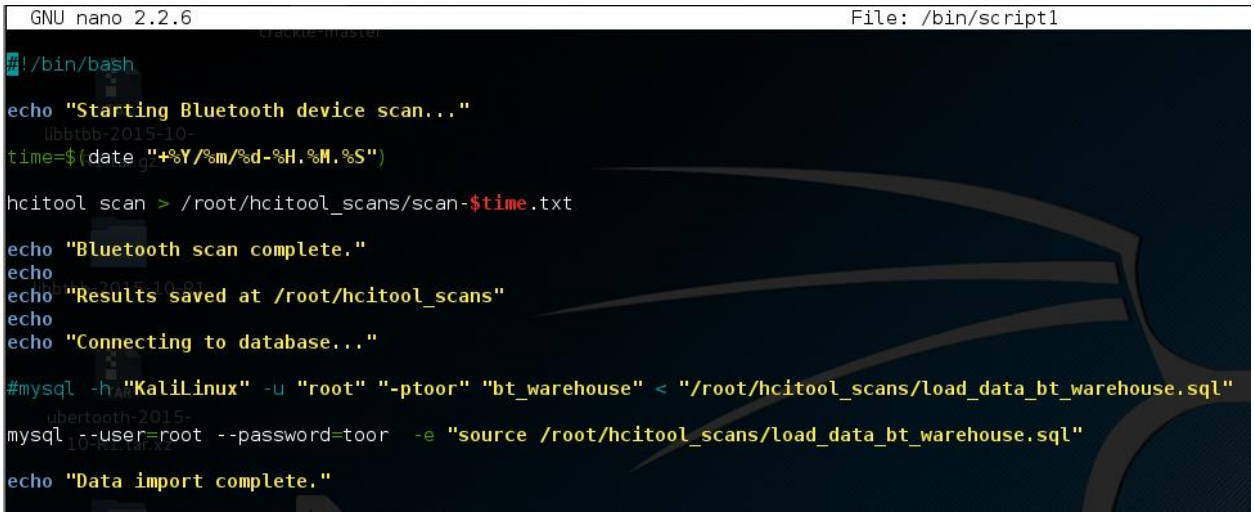


Figure 11. Shows the logic behind the ToothScraper tool.

Ubertooth One

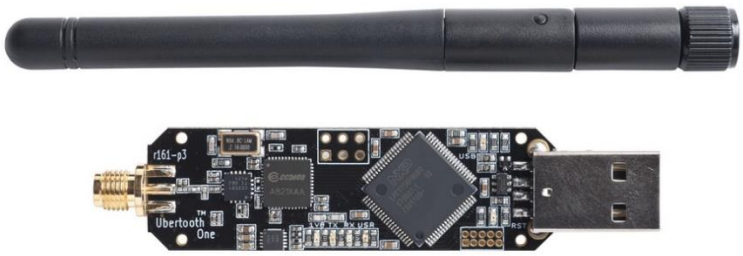


Figure 12. Displays a picture of an Ubertooth One and its antenna.

Figure 10 shows an Ubertooth One, which is a 2.4 GHz wireless radio device with a detachable antenna that is capable of sniffing Bluetooth packets. In our initial demonstration, we used this device to display real-time Bluetooth communication and capture a conversation between two devices – an Apple iPhone 6 and a wireless Bluetooth keyboard. Using the command "ubertooth-rx", the Ubertooth will display a stream of packets once it detected Bluetooth activity. The presence of a piconet is indicated by the same LAP showing up after an "ubertooth-rx" scan. Next we used the LAP of the iPhone 6 and modified the original command to "ubertooth-rx –l", which allows the Ubertooth to find the UAP of the device, based off parts of the internal clock value. Figure 10 represents an example of this command.

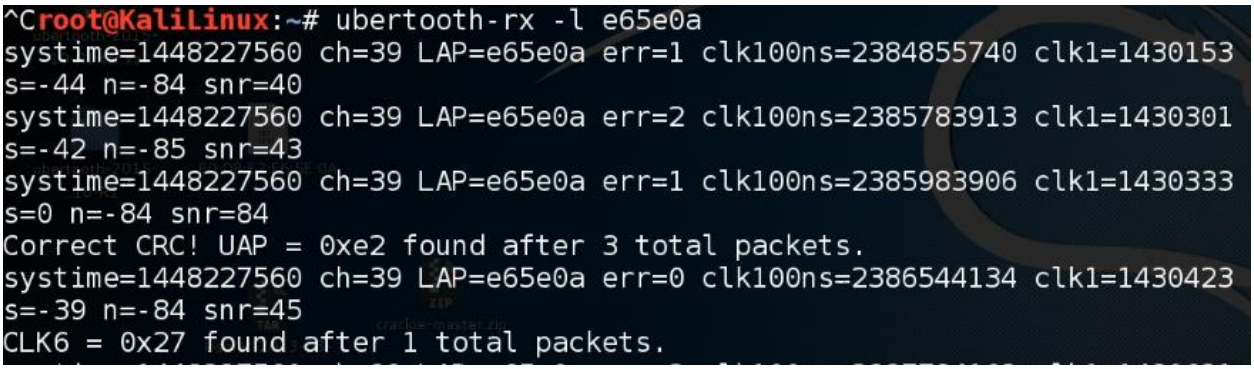


Figure 13. An Ubertooth One command that scans a piconet to determine a device's 8-bit UAP.

These results were saved to a .pcap file to be viewable in Wireshark. Figure 12 shows the same command being run, but this time with the addition of a switch to direct the results to a file.

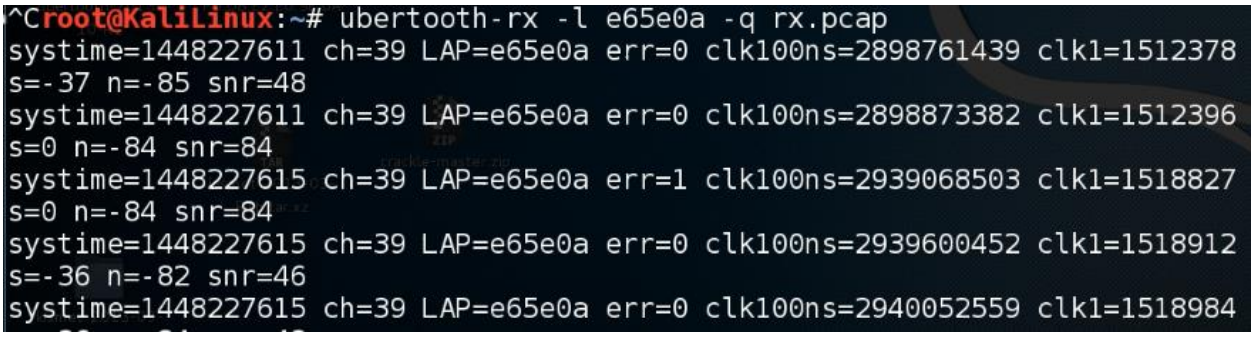


Figure 14. An Ubertooth One command that scans a piconet to determine a device's 8-bit UAP, and saves the output to a .pcap file named "rx".

The "ubertooth-btle" command allows the Ubertooth One to scan for Bluetooth LE packets, with the –p switch designating promiscuous mode (allows the Ubertooth One to pick up all packets transmitted in its range) and the –c switch allowing the user to save the output to a file, in our case a .pcap file. Figure 14 shows the results of running this command:

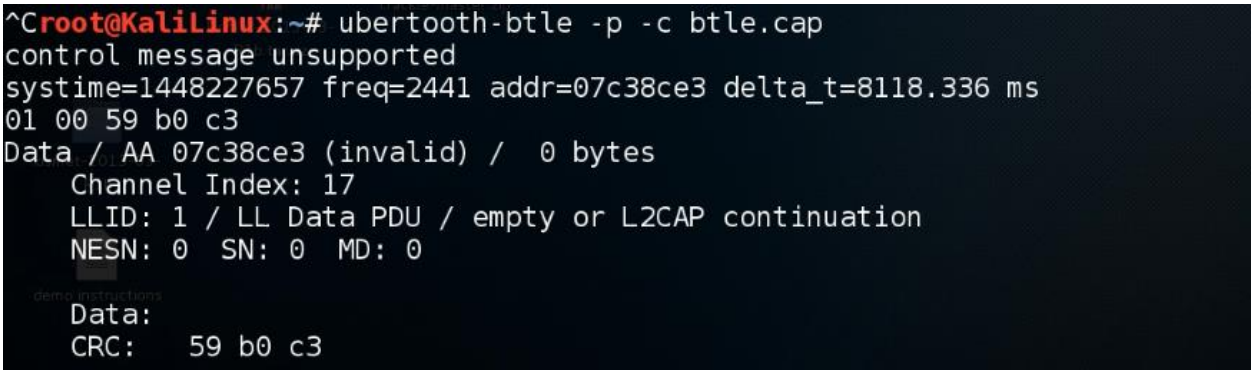


Figure 15. An Ubertooth One command that shows a stream of Bluetooth LE traffic in real-time.

Now that these results have been saved to .pcap files, we can open them in Wireshark. Our purpose of using Wireshark to view Bluetooth LE packets was to capture a pairing sequence between the iPhone 6 and the wireless keyboard, so that the TK could be captured and brute forced with a tool called Crackle. The data captured was encrypted because the two devices authenticated during the pairing session, therefore encrypting any future communications. Figure 15 displays the Bluetooth LE packets captured during the command "ubertooth-rx –l e65oa -q rx.pcap".

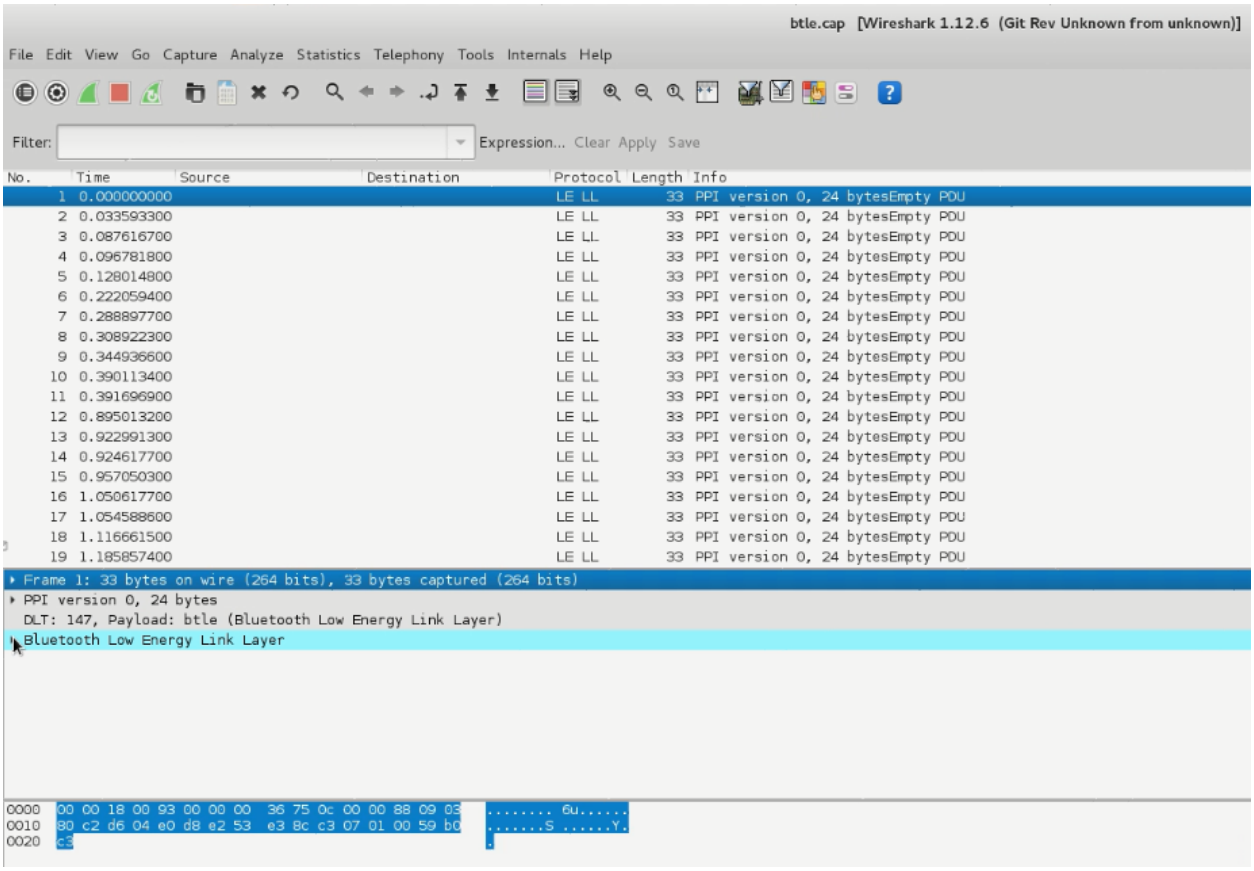


Figure 16. A screenshot of the rx.pcap file in Wireshark.

Kali Linux

Kali Linux is a Linux-based vulnerability and penetration testing platform that contains pre-installed tools conducting reconnaissance, scans, enumeration, exploitation, and reporting tools in one distribution. We ran this operating system in a virtual machine using VMware Workstation 11.1.1. We chose to use Kali Linux based on its pre-installed tools pertaining specifically to Bluetooth technology that would allow us to scan for devices, retrieve detailed information, and save those outputs to files for later use.

btscanner

btscanner is a GUI driven application that can perform inquiry and brute force scans for Bluetooth devices. Once the application detects discoverable devices, it allows the user to select a device to view advanced information such as the device's manufacturer, its BD_ADDR, last seen and first seen timestamps, services, and its clock. Btscanner was used in our initial demonstration to show how a device transmits its device name and BD_ADDR to other devices that send out inquiry packets. Figure 16 shows the interface of btscanner and its configurable options.

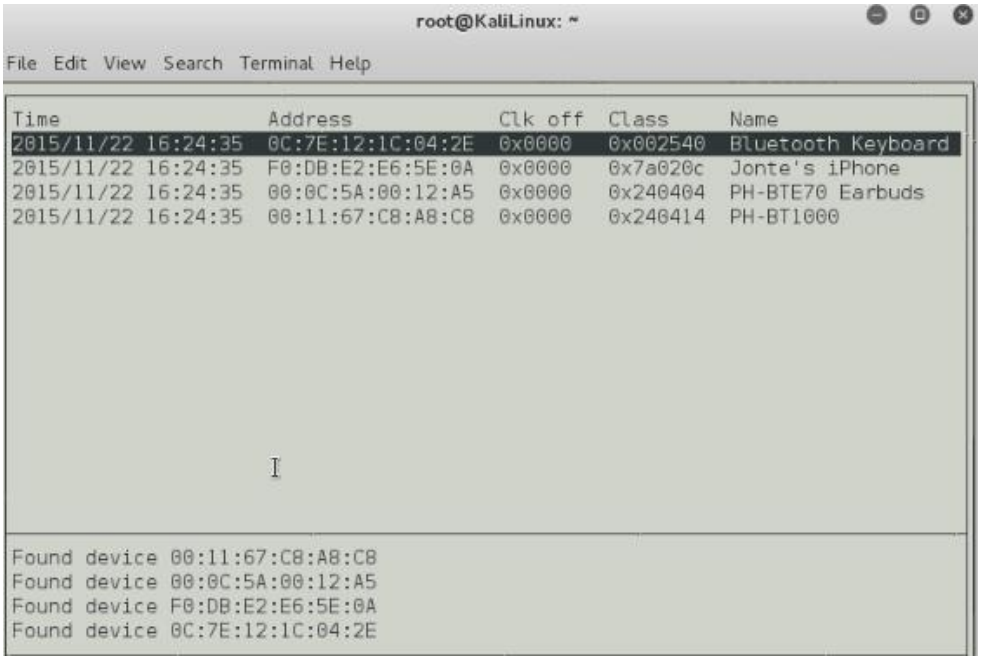


Figure 17. Shows the application "btscanner" scanning for discoverable devices and listing them in a GUI interface.

Hcitool

Hcitool is an application that can scan discoverable devices through inquiry or periodic scans, with additional capabilities of the link quality and power level of a device. This

application shares similar functionalities in comparison to btscanner, but hcitool appears to have a more diverse set of options for handling Bluetooth devices. Figure 17 shows the output of an inquiry scan using hcitool.

Figure 18. Shows the application “hcitool”, its output from an inquiry scan, and a detailed device info request.

Wireshark

Wireshark is a network packet analyzer that allows a user to inspect a group of packets in detail. The use of Wireshark in the initial demonstration was to show that the Ubertooth One successfully captured Bluetooth communications by calculating the hopping sequence and following the connection.

MySQL

MySQL is an open source database management system. We used MySQL to store the output of hcitool scans, as a way to centralize the BD_ADDR and device names to be placed into a structurally organized management system.

phpMyAdmin

phpMyAdmin is an open source tool used to assist in the maintenance of a MySQL database through a web interface. We used this tool in conjunction with our MySQL database to better visualize our data and determine the best way to store Bluetooth device information. Figure 18 shows an early screenshot of our database, "bt_warehouse", using phpMyAdmin to authenticate to our MySQL database and display its contents.

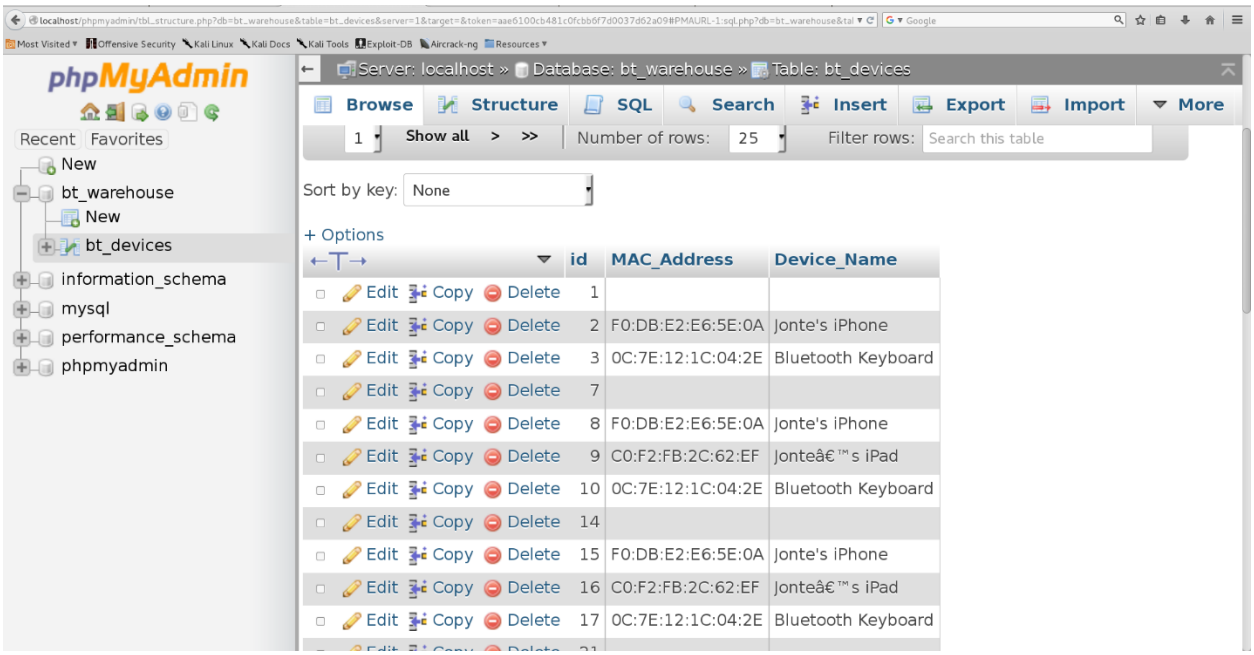


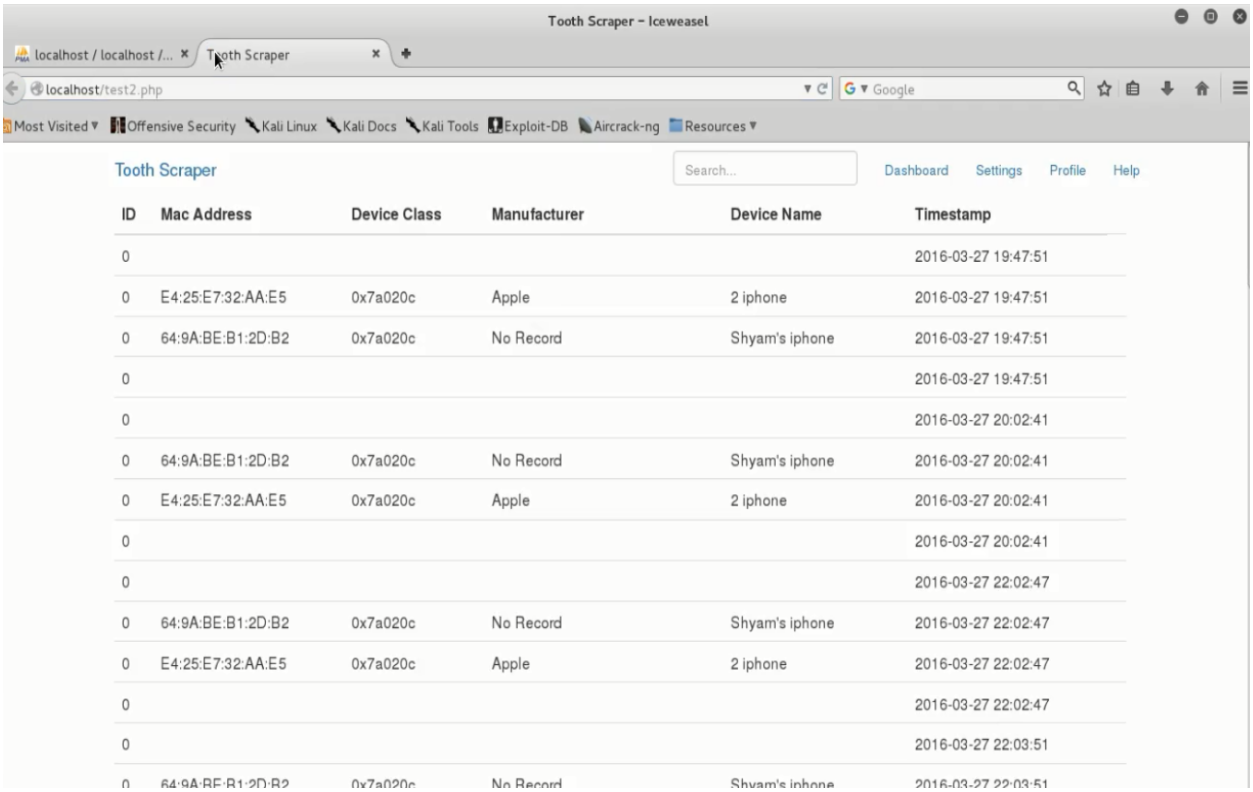
Figure 19. Displays the contents of table "bt_devices" within the "bt_warehouse" database.

After researching various Bluetooth vulnerabilities, we came to the conclusion that we would focus on gaining a clearer understanding of Bluetooth as a technology, and leave the

implementation of exploiting target devices for the second semester. This allows us to troubleshoot any problems that arise with increased efficiency due to our research knowledge. Instead of demonstrating an exploit in the fall semester, we decided to showcase the capabilities of the Ubertooth One in conjunction with Kali Linux to sniff Bluetooth packets and display them in Wireshark. The main problem we encountered in building the demonstration was configuring the Ubertooth One inside the Kali Linux VM. In order to use the command that allows the Ubertooth One to sniff for low energy devices and capture a pairing request, we first needed to update the firmware on the Ubertooth One. Unfortunately, upon putting the device into DFU mode, it lost its connection from the VM and became and unrecognized USB device.

Bootstrap

Bootstrap is a front-end framework that can be used to customize a web sites functionality and visual appeal. We used an existing Bootstrap template to connect to our database and display its contents in a visually appealing manner. Figure 20 below displays our Bootstrap web site.



ID	Mac Address	Device Class	Manufacturer	Device Name	Timestamp
0					2016-03-27 19:47:51
0	E4:25:E7:32:AA:E5	0x7a020c	Apple	2 iphone	2016-03-27 19:47:51
0	64:9A:BE:B1:2D:B2	0x7a020c	No Record	Shyam's iphone	2016-03-27 19:47:51
0					2016-03-27 19:47:51
0					2016-03-27 20:02:41
0	64:9A:BE:B1:2D:B2	0x7a020c	No Record	Shyam's iphone	2016-03-27 20:02:41
0	E4:25:E7:32:AA:E5	0x7a020c	Apple	2 iphone	2016-03-27 20:02:41
0					2016-03-27 20:02:41
0					2016-03-27 22:02:47
0	64:9A:BE:B1:2D:B2	0x7a020c	No Record	Shyam's iphone	2016-03-27 22:02:47
0	E4:25:E7:32:AA:E5	0x7a020c	Apple	2 iphone	2016-03-27 22:02:47
0					2016-03-27 22:02:47
0					2016-03-27 22:03:51
0	64:9A:BE:B1:2D:B2	0x7a020c	No Record	Shyam's iphone	2016-03-27 22:03:51

Figure 20. Displays the contents of our database using a Bootstrap template.

Due to the complications in updating the firmware, we decided to capture Bluetooth packets that were communicated in a pairing session between an iPhone 6 and a wireless Bluetooth keyboard. These packets were then saved to .pcap files, so that we could inspect the data inside each packet in Wireshark. Viewing these files in Wireshark showed us that the packets were encrypted, therefore any further information was unable to gather from the capture files. Wireshark did interpret and display the Bluetooth Low Link Energy Layer packets correctly, due to previous configuration commands that were executed prior to the demonstration.

During the demonstration, we also decided to showcase the promiscuous scanning feature of the Ubertooth One. By executing the command “ubertooth-rx”, the Ubertooth One was able to

display all Bluetooth packets that it could capture, whether its intended recipient was the scanning device or not. The “ubertooth-rx” command was executed again, this time filtering out the packets to those that belonged to the iPhone 6. After about five seconds of capturing packets, the Ubertooth One software was able to determine the first two characters that preceded the LAP (Lower Address Part) of the iPhone’s BD_ADDR address. Following this command, Bluetooth traffic was generated purposely in order to see Bluetooth communicating in real time between two devices.

For the second semester, we developed two deliverables. The first deliverable was a tool that can scan for discoverable Bluetooth devices and store device metadata in a database automatically. The second deliverable was a security framework that provides recommendations for securing Bluetooth technology.

Bibliography

1. Accessed November 3, 2015. https://www.nsa.gov/ia/_files/factsheets/i732-016r-07.pdf.

2. Accessed November 3, 2015. https://www.sans.org/reading-room/whitepapers/wireless/bluetooth-inherent-security-issues-945.

3. Accessed November 3, 2015. http://www.airccse.org/journal/ijdps/papers/0112ijdps10.pdf.

4. “A menu of Bluetooth attacks,” GCN, Accessed October 19, 2015, https://gcn.com/articles/2005/07/20/a-menu-of-bluetooth-attacks.aspx , entire page.

5. Authy Makes Using Two-Factor Authentication." TechCrunch. Accessed February 12, 2016. http://techcrunch.com/2013/07/31/authy-makes-using-two-factor-authentication-easier-by-connecting-your-phone-and-mac-over-bluetooth/.

6. "BackTrack Linux - Penetration Testing Distribution." BackTrack Forums RSS. Accessed November 3, 2015. http://www.backtrack-linux.org/forums/showthread.php?t=5637.

7. “Bluetooth Security – BluetoothSecurity.pdf.” Accessed March 1, 2016. http://www.ece.umd.edu/class/ents650/BluetoothSecurity.pdf.

8. "Bluetooth SIG and FIDO Alliance Deliver Two-factor Authentication Via Bluetooth Smart." 2016. Accessed March 17, 2016. https://www.bluetooth.com/news/pressreleases/2015/07/15/bluetooth-sig-fido-alliance-deliver-two-factor-authentication-via-bluetooth-smart.

9. “bluetooth-technology-by-polite-group-10-638.jpg.” Accessed March 1, 2016. http://image.slidesharecdn.com/bluetoothtechnologybypolitegroup-130422023230-phpapp01/95/bluetooth-technology-by-polite-group-10-638.jpg?cb=1366598108.

10. “crackle, crack Bluetooth Smart (BLE) encryption.” Accessed March 1, 2016. https://lacklustre.net/projects/crackle/.

11. "Greatscottgadgets/ubertooth." GitHub. Accessed December 3, 2015. https://github.com/greatscottgadgets/ubertooth/wiki/Build-Guide.

12. "Hack Brief: Upgrade to IOS 9 to Avoid a Bluetooth IPhone Attack." . Accessed February 18, 2016. http://www.wired.com/2015/09/hack-brief-upgrade-ios-9-now-avoid-bluetooth-iphone-attack/.

13. "Hcitool." Hcitool. Accessed November 3, 2015. http://linuxcommand.org/man_pages/hcitool1.html.

14. "How To Hack Phones Bluetooth With Kali Linux And Backtrack - Hack Training." Hack Training. August 16, 2014. Accessed November 3, 2015. http://hack.training/hack-phones-bluetooth-kali-linux-backtrack/.

15. “Introduction to Bluetooth Low Energy Security.” Accessed March 1, 2016. http://twelvedot.com/blog/?p=621.

16. “Itl Bulletin for August 2012 Security of Bluetooth Systems and Devices: Updated Guide Issued by the National Institute of Standards and Technology (Nist).” Accessed March 1, 2016. http://csrc.nist.gov/publications/nistbul/august-2012_itl-bulletin.pdf.

17. "Mikeryan/crackle." GitHub. Accessed November 3, 2015. https://github.com/mikeryan/crackle.

18. “Nexus 7: A hackers toolkit. Part 1 - Setup and hacking WPA networks,” Intohand, Accessed October 19, 2015, http://intohand.com/blog/post/nexus-7-a-hackers-toolkit.-part-1-setup-and-hacking-wpa-networks , entire page.

19. "NIST Issues Guide to Fixing Bluetooth Vulnerabilities -- GCN." NIST Issues Guide to Fixing Bluetooth Vulnerabilities -- GCN. Accessed November 3, 2015. https://gcn.com/articles/2012/06/13/nist-bluetooth-security-guide.aspx.

20. "Now I Wanna Sniff Some Bluetooth: Sniffing and Cracking Bluetooth with the UbertoothOne." RSS. Accessed November 3, 2015. http://www.security-sleuth.com/sleuth-blog/2015/9/6/now-i-wanna-sniff-some-bluetooth-sniffing-and-cracking-bluetooth-with-the-ubertoothone.

21. “olzak_bluetooth_table2.jpg.” Accessed March 1, 2016. http://tr1.cbsistatic.com/hub/i/2015/06/03/0de8c373-0989-11e5-940f-14feb5cc3d2a/olzak_bluetooth_table2.jpg.

22. "Security of Bluetooth Low Energy (BLE) Link-layer Encryption." - Information Security Stack Exchange. Accessed November 3, 2015. http://security.stackexchange.com/questions/100443/security-of-bluetooth-low-energy-ble-link-layer-encryption.

23. "Sniffing/logging Your Own Android Bluetooth Traffic." - Stack Overflow. Accessed November 3, 2015. http://stackoverflow.com/questions/23877761/sniffing-logging-your-own-android-bluetooth-traffic.

24. “sp800-121_rev1.pdf.” Accessed March 1, 2016. http://csrc.nist.gov/publications/nistpubs/800-121-rev1/sp800-121_rev1.pdf.

25. “ubertooth-one-052-1024.jpg.” Accessed March 1, 2016. http://hackerwarehouse.com/wp-content/uploads/2012/12/ubertooth-one-052-1024.jpg.

26. "What Can an Attacker Do with Bluetooth and How Should It Be Mitigated?" Windows. Accessed November 3, 2015. http://security.stackexchange.com/questions/26356/what-can-an-attacker-do-with-bluetooth-and-how-should-it-be-mitigated.

27. “What Is Bluejacking?.” Accessed March 1, 2016. http://www.tech-faq.com/bluejacking.html.

28. “5216816c757b7f1f668b4567.png.” Accessed March 1, 2016. https://cdn.sparkfun.com/assets/a/e/a/b/5/5216816c757b7f1f668b4567.png.