Regular Paper

Communications of the CCISA
Vol. 25 No. 1 Feb. 2019

低功耗藍芽協定安全模糊測試框架

林聖翔[1]、卓信宏[2]、陳麒元[3*]、李育杰[4]
[123]國立宜蘭大學資訊工程學系、[4]安華聯網科技
[1]p8989p72@gmail.com、[2]Hsin-Hung@ieee.org、[3]chiyuan.chen@ieee.org、
[4]jackyli@onwardsecurity.com

摘要

低功耗藍芽(Bluetooth Low Energy，BLE)由於其省電的特性，許多行動裝置及穿戴裝置皆支援低功耗藍芽通訊技術，加上近年物聯網相關應用的普及，越來越多個人資料透過低功耗藍芽通訊協定來進行傳輸，然而針對各種藍芽技術的攻擊手法層出不窮，如何檢測低功耗藍芽裝置的安全性成為急需克服的挑戰。本研究採用軟體測試中常見的黑箱測試方法-模糊測試(Fuzz Testing)，提出一低功耗藍芽協定安全模糊測試框架，並且採用開源的軟硬體資源實作測試平台，進一步分析進行低功耗藍芽協定測試所遭遇的困難與解決方案。

關鍵詞：低功耗藍芽、模糊測試、物聯網

Security Fuzz Testing Framework for Bluetooth Low Energy Protocols

Sheng-Xiang Lin[1], Hsin-Hung Cho[2], Chi-Yuan Chen[3*], Yu-Chieh Li[4]
[123]Department of Computer Science and Information Engineering, National Ilan University
[4]Onward Security
[1] p8989p72@gmail.com, [2] Hsin-Hung@ieee.org, [3]chiyuan.chen@ieee.org,
[4]jackyli@onwardsecurity.com

Abstract

Due to the power saving feature of Bluetooth Low Energy (BLE), many mobile devices and wearable devices support BLE communication technology. In recent years, the popularity of IoT related applications, more and more personal data transferred through the BLE protocol. However, there are various attack techniques for Bluetooth technologies. How to test the security of BLE devices has become an urgent challenge to overcome. In this paper, we utilized the black box test method, Fuzz Testing, which is common in software testing. This paper presents a Security Fuzz Testing Framework for BLE Protocols and uses open

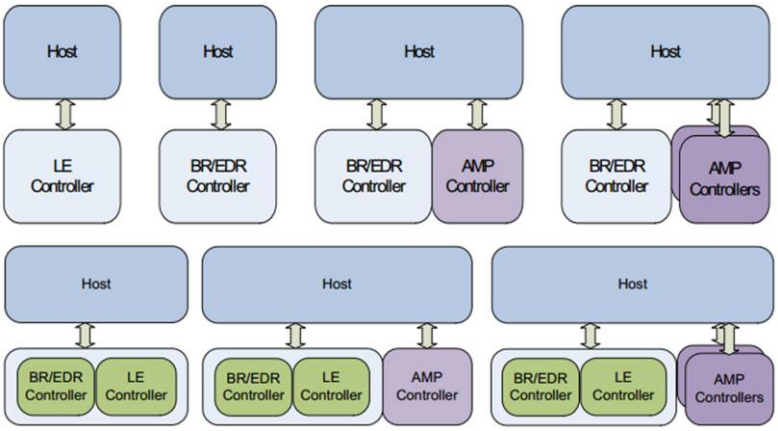
* 通訊作者 (Corresponding author.)

source hardware/software resources to implement the testing platform. We also analyze the difficulties and solutions encountered in the testing of BLE protocols.

Keywords: Bluetooth Low Energy, Fuzz Testing, Internet of Things

壹、前言

藍芽技術的發展已有二十年，在 1999 年公布了 1.0 版，而後做了幾次更新，手機也開始搭載了藍芽技術，2004 年 2.0 版附加了 EDR(Enhanced Data Rate)的功能，此時也已經出現許多讓人感到驚奇的攻擊手法，2.0+EDR 的版本已於 2014 年由 Bluetooth SIG 公告作廢及撤回，隨後於 2007 年更新 2.1+EDR 版本。隨著物聯網的蓬勃發展，藍芽技術也跟上了腳步，於 2010 年推出了 4.0 版本，除了修補許多存在的漏洞，並將之前所有的版本做了一個整合，引入了低功耗(LE，Low Energy)的技術，因此藍芽 4.0 有了多種模式，這取決於藍芽晶片的功能被設計成什麼類型，圖一出自 Bluetooth Protocol Spec [1]，說明了藍芽架構中 Host 與 Controller 之間的獨立性，並列出 4.0 版本中所有種類的藍芽晶片，有些晶片支援 LE，有些則支援 BR/EDR，還有些還支援了音頻協議 AMP。



圖一：Bluetooth Host and Controller Combinations [1]

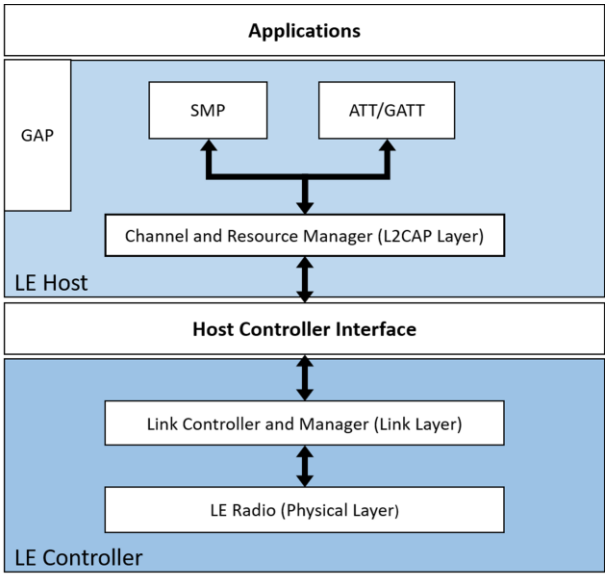
因為低功耗藍芽(Bluetooth Low Energy，BLE)省電的特性，並且大部分手機皆搭載了藍芽技術，許多藍芽穿戴式裝置在市場逐漸普及。藍芽架構上極大的變化，導致後來針對藍芽的攻擊手法已經鮮少人在討論，雖然新發展出的低功耗藍芽存在著幾點攻擊上的困難，但是為了保持低功耗省電，在架構上趨向於簡單且體積小的特性，讓整體架構存在許多安全上的隱憂。

在本文中，我們提出了一個低功耗藍芽協定安全模糊測試框架，並且透過結合一些藍芽相關的開源軟硬體工具，建構一個低功耗藍芽協定安全模糊測試平台，並在文中探

討為何至今鮮少有關於藍芽模糊測試相關的研究，其中有幾個測試上的困難點。在文獻探討章節中，我們首先將簡單說明低功耗藍芽協議上的架構，探討低功耗藍芽的低安全性，以及現今與模糊測試和攻擊手法相關的文獻。接著則提出藍芽模糊測試平台，並劃定測試範圍，分別說明每個元件的功能以及可以利用的工具，最後為本文的結論。

貳、文獻探討

藍芽版本眾多，圖二將低功耗藍芽的架構從 Spec [1]中獨立出來，讀者也能從[2]一書中了解更多關於低功耗藍芽的架構。



圖二：低功耗藍芽架構圖

本文將測試範圍設定在低功耗藍芽上，架構僅有 LE Host 跟 LE Controller，各協議層內容如圖二。當接收端收到藍芽訊號時，藍芽訊號會在 Physical Layer 進行訊號解調變，並將類比訊號轉成數位訊號後往上送至 Link Layer。Link Layer 除了會將被 Scramble 的封包解回來，還會將原有的 header 丟棄，並加上 Host Controller Interface (HCI)專屬的 header，以透過 HCI 了解這個封包屬於什麼類型，以及告訴 L2CAP Layer 該如何處理這個封包。

Controller 的獨立性導致我們並無法從系統中，了解 Controller 運行狀況，只有當封包已在 Host 內才能夠控制。L2CAP Layer 負責管理封包要往哪個 Layer 送，Security Manager Protocol (SMP)負責藍芽通訊間的加密，Generic Attribute Profile (GATT)資料結

構定義了兩個設備間傳輸的資料為何種服務，而資料的內容則是各廠商自行定義，定義內容皆撰寫於 Applications 層，Attribute Layer (ATT)則使用 GATT 資料來組織封包，並且規範資料該如何被處理，於[2]一書中甚至還提到，藍芽架構的縮減，讓原本複雜的協議層濃縮成 ATT 協議。

因此目前看到針對 BLE 設備的攻擊，皆是由於 ATT 協議過於簡單而造成的漏洞，於[3]論文中，正是透過分析兩設備間 ATT 協議的封包內容，來將廠商定義的欄位名稱分析出來，我將此攻擊的攻擊面定義在藍芽設備間的連線。除了這類利用 ATT 協議缺陷的攻擊之外，在[4]的白皮書中，提到針對藍芽的中間人攻擊，透過佈署藍芽假節點，誘使目標連接後，就能竊聽設備間的資料傳輸。藍芽設備分為中央設備(Central Device)和外圍設備(Peripheral Device)，外圍設備在尚未連線時會不斷地廣播自身的訊息，而文中藍芽假節點是利用以 Nodejs 撰寫而成的 bleno[5]作為構建 Bluetooth Protocol Stack 之外圍設備的主要工具。此外，藍芽設備的攻擊面還有無線電訊號，無線電訊號就需要以特定的射頻(RF)裝置來監聽無線電訊號，於[6]的論文中，就是以無線電的角度來分析 BLE 的安全性，以及攻擊的難度。在 2.4GHz 中，存在著許多種類的無線訊號，Wifi 和藍芽就是其中之一，而藍芽為了增強訊號的穩定性，採用了自適應跳頻的技術，在連線後除了每隔一段時間就會跳固定間距的頻率進行傳輸之外，當偵測到某頻率訊號過於雜亂，也會改由其他頻率進行傳輸。這造就攻擊者必須從封包中了解跳頻規則，並控制 RF 設備來做跳頻跟蹤，這也是為何無線電針對藍芽攻擊的困難性高。而該篇論文，展現了如何竊聽到藍芽連線封包，甚至提供了第一款藍芽射頻監聽設備 Ubertooth 來跟蹤連線的跳頻，除此之外該工具實現了藍芽廣播封包的偽造[7]。

攻擊者常用的 RF 裝置為軟體定義無線電(Software Defined Radio，SDR)設備，能夠透過軟體來改變 RF 的頻率，進而產生任意頻率的無線電訊號，甚至能將訊號進行錄製，來達到重放攻擊。 [8]中的工具還是以 SDR 設備來進行訊號的監聽，除了具備[7]的功能外，具備更完善的功能，能夠偽造任何類型的藍芽封包，[2]中也提及了各個藍芽封包類型的用途，本文將各類型封包整理於下表一中。

表一: 藍芽封包類型

Type	Layer	Packet name
廣播	Link	ADV_IND
		ADV_DIRECT_IND
		ADV_NONCONN_IND
		ADV_SCAN_IND
		SCAN_REQ
		SCAN_RSP
		CONNECT_REQ
資料	ATT	ATT
	SMP	SMP
控制	L2CAP	LL_CONNECTION_UPDATE_REQ

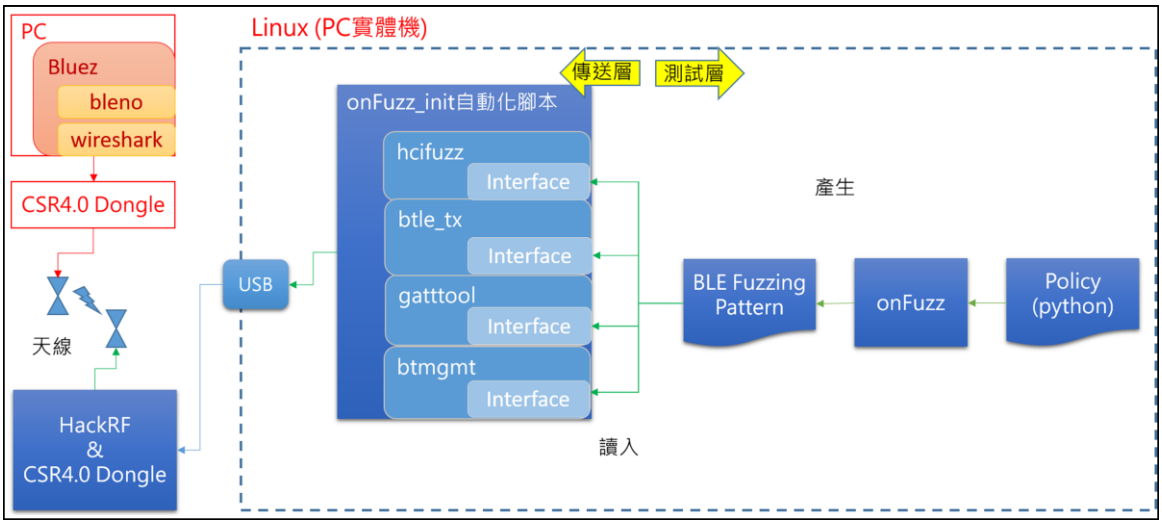
		LL_CHANNEL_MAP_REQ
		LL_TERMINATE_IND
		LL_ENC_REQ
		LL_ENC_RSP
		LL_START_ENC_REQ
		LL_START_ENC_RSP
		LL_FEATURE_REQ
		LL_FEATURE_RSP
		LL_PAUSE_ENC_REQ
		LL_PAUSE_ENC_RSP
		LL_VERSION_IND
		LL_REJECT_IND

在設備傳送或接收 CONNECT_REQ 前，皆是進行封包的廣播，收到 CONNECT_REQ 後，兩者設備便建立連線，以及開始跳頻，這時候才會傳輸資料封包和控制封包。

在藍芽協議的模糊測試方面，[6]文中提到他們提供的 POC 封包注入器，為未來的主動式攻擊奠定了基礎，其提出理論性的攻擊手法，論述模糊測試是可行的。在[9]的白皮書中，展示藍芽協議的模糊測試，只是與本文不同的是，文中展現的是車載藍芽系統的模糊測試，測試的範圍包括 L2CAP、A2DP(Advenced Audio Distribution Profile，藍芽音頻傳輸模型拹定)，以及 HFP (Hands-Free Profile，免手持裝置規範)。而在[10]一文中，展示了一個針對 BLE 的安全性測試框架，並且分別針對 MITM、Flooding 及 Fuzzing 進行測試，但其中 Fuzzing Attack 僅針對 GATT Server Profile 進行測試。

參、低功耗藍芽協定安全模糊測試框架

本研究的模糊測試框架與實作平台以受測端的外圍設備、Fuzzer、Monitor 構成，外圍設備我們選擇 bleno 來架設，如圖三、圖四及表二所示。相比於另外一套外圍設備軟體 Android-BlePeripheral 來說，bleno 沒有任何硬體上的限制，故最為方便且合適，Fuzzer 主要透過 SDR 裝置 HackRF 作為廣播封包的測試，以 CSR4.0 USB 藍芽接收傳輸器來做資料和控制封包的測試。HackRF 透過軟體無線電的方式，模擬藍芽訊號的頻率，以 btle_tx[8]軟體來填充封包和封包編碼，CSR4.0 USB 藍芽接收傳輸器則是使用 Linux 開源 Bluetooth Protocol Stack 平台 Bluez-5.43，建構完整的藍芽協議架構，並組織封包來傳送。而 Monitor 則以 Ubertooth[7]來跟蹤藍芽訊號的跳頻，並監聽封包內容。



圖三：低功耗藍芽協定安全模糊測試框架

圖四：低功耗藍芽協定安全模糊測試平台

表二:測試軟硬體設備需求

角色	硬體	軟體
Fuzzer	HackRF	btle_tx
	PC	4.9.0-kali3-amd64
DUT/Victim	CSR4.0 USB Dongle	Bleno、Bluez-5.43
	PC	4.9.0-kali3-amd64
Monitor	CC2540 USB Dongle	TI SmartRF Protocol Packet Sniffer
	Ubertooth One	ubertooth-btle
	PC	Windows 7 或 4.9.0-kali3-amd64(擇一)

低功耗藍芽架構中分成三個部分，從上到下為 Host、HCI、Controller，在模糊測試的流程中，扮演 Host 的是 Ubuntu 主機，透過藍芽協議堆疊 bluez 完成 Host 中 ATT、L2CAP 等協議的功能。而扮演 Controller 的則是 CSR4.0 USB Dongle，Dongle 內的藍芽晶片和天線則完成了 Link Layer 和 Pysical Layer 等功能，最後一個則為 HCI，它是 Host 和 Controller 的傳輸的管道，當 Host 開始發送 Fuzzing Pattern 時，會從 L2CAP 之上的 Layer，將 Pattern 填入封包中的 Payload，然後加上 HCI 層的 header，經過封裝後才發給 Controller。這時 Controller 接收到來自 Host 的 HCI 命令，會去識別 header 和 Payload 的內容，然後將 Payload 取出與 Link Layer 的 header 封裝起來，才由 Pysical Layer 經調變發送出去。

圖三中的自動化腳本內分成四個部分，這四個部分分別對應到測試的協議層或封包格式，對應的內容如表三所示。Bluez 為開源的 Bluetooth Protocol Stack 平台，最常被 Linux 平台所用， Bluez 已經能夠幾乎完整實作藍芽架構，我們進而修改 Bluez 所提供的工具 hcitool 成 hcifuzz，加入傳入 Pattern 的參數，並整合封包發送的函式，指定 Bluez 來發送我們的模糊測試封包，參數設置方式與執行範例如圖五。

表三: 自動化腳本與測試接口對應

接口	程式
Advertisement Packet	btle_tx
ATT Packet	gatttool
SMP Packet	btmgmt
CONNECT_REQ	hcifuzz
LL_CONNECT_UPDATE_REQ	
LL_VERSION_IND	
LL_FEATURE_REQ	
LL_TERMINATE_IND	

```
hcifuzz - HCI Tool ver 5.43
Usage:
        hcifuzz [options] <command> [command parameters]
Options:
        --help  Display help
        -i dev  HCI device
        -p ptn  fuzzing pattern
Commands:
        m05     CONNECT_REQ
        m06     LL_CONNECTION_UPDATE_REQ
        m07     LL_VERSION_IND
        m08     LL_FEATURE_REQ
        m09     LL_TERMINATE_IND
Example:
        sudo hcifuzz -i hci0 -p 1900040004000F000F00000C8000010001 m05 00:1A:7D:DA:71:11
        sudo hcifuzz -i hci0 -p 0c0C80096000070C8000010001 m06 00:1A:7D:DA:71:11
        sudo hcifuzz -i hci0 -p 02 m07 00:1A:7D:DA:71:11
        sudo hcifuzz -i hci0 -p 02 m08 00:1A:7D:DA:71:11
        sudo hcifuzz -i hci0 -p 0313 m09 00:1A:7D:DA:71:11

For more information on the usage of each command use:
        hcifuzz <command> --help
```

圖五：hcifuzz 參數設置說明與範例

此外 btle_tx 為低功耗藍芽無線發射器，雖然它已經能直接自訂封包了，但是 header 的長度欄位不正確會造成整個封包被丟棄，因此我們需要讓它自動校正長度欄位的隨機數值，程式碼如圖六，bit_out 陣列放置封包所有內容，透過陣列中每一個 bit 做邏輯上的位移，來賦予並且修正數值。

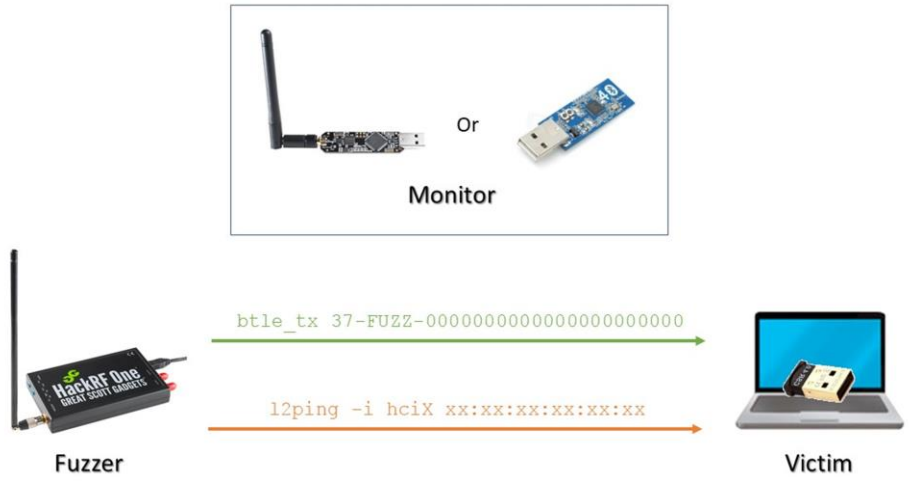
```
void fill_fuzz_length(int payload_len, char *bit_out) {
        bit_out[8] = 0x01&(payload_len>>0);
        bit_out[9] = 0x01&(payload_len>>1);
        bit_out[10] = 0x01&(payload_len>>2);
        bit_out[11] = 0x01&(payload_len>>3);
        bit_out[12] = 0x01&(payload_len>>4);
        bit_out[13] = 0x01&(payload_len>>5);
}
fill_fuzz_length(payload_len, pkt->info_bit+5*8);
```

圖六：校正封包長度欄位

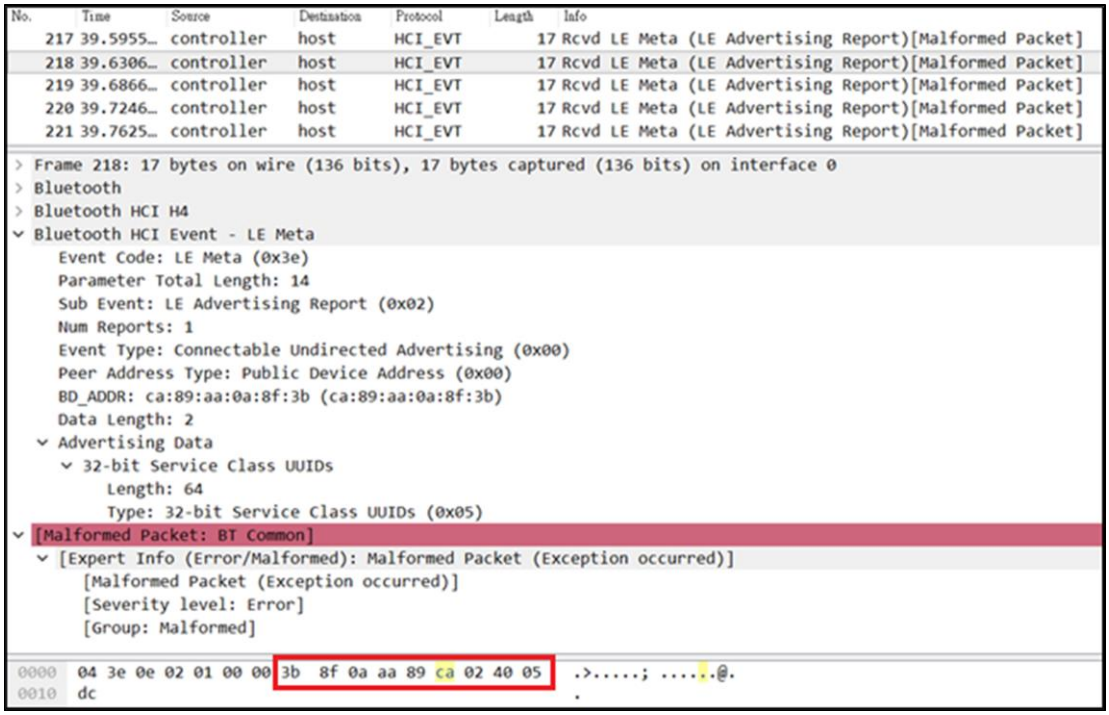
肆、低功耗藍芽協定安全模糊測試

根據圖二的 BLE 架構，協定安全模糊測試可以分為兩種方式。第一種測試是在沒有連接的情況下，對兩個設備間的連線進行做模糊測試，我們稱作為無線電無連線測試，透過軟體定義無線電來偽造正常的封包注入，測試範圍以跳頻前後和封包類型做區分，分為廣播頻道和資料頻道，封包類型如表一。第一種測試如圖七所示，我們利用 Bleno 來搭建外圍設備，透過安華聯網科技的 onFuzz 產生大量要測試的 Pattern，再交由 HackRF

的 btle_tx 生成 CRC，一筆一筆發送至目標端做測試，再發送 L2CAP 封包，確認目標正常運作與否。除此之外 DUT/Victim 設備在發送廣播封包時，必須進入掃描狀態，而資料封包或控制封包，必須與另外一個藍芽設備進入連接狀態。圖八為對廣播封包中，包括 header 共 6 到 37 個 bytes 進行測試的範例。

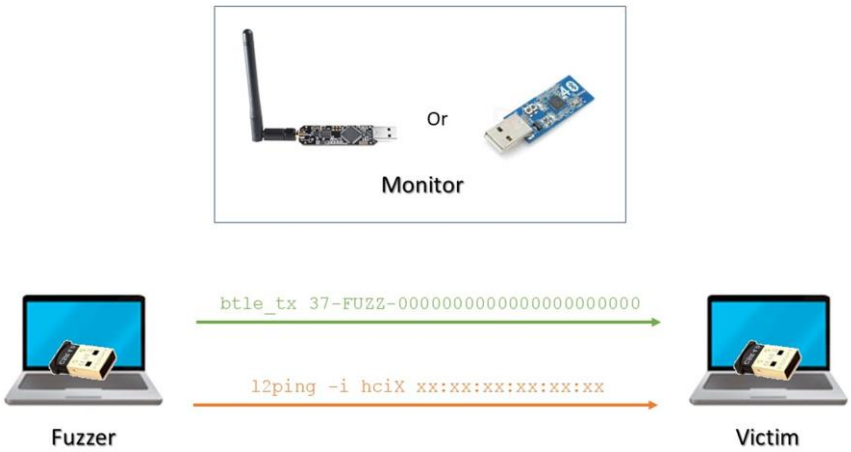


圖七：無線電訊號無連線測試架構示意圖

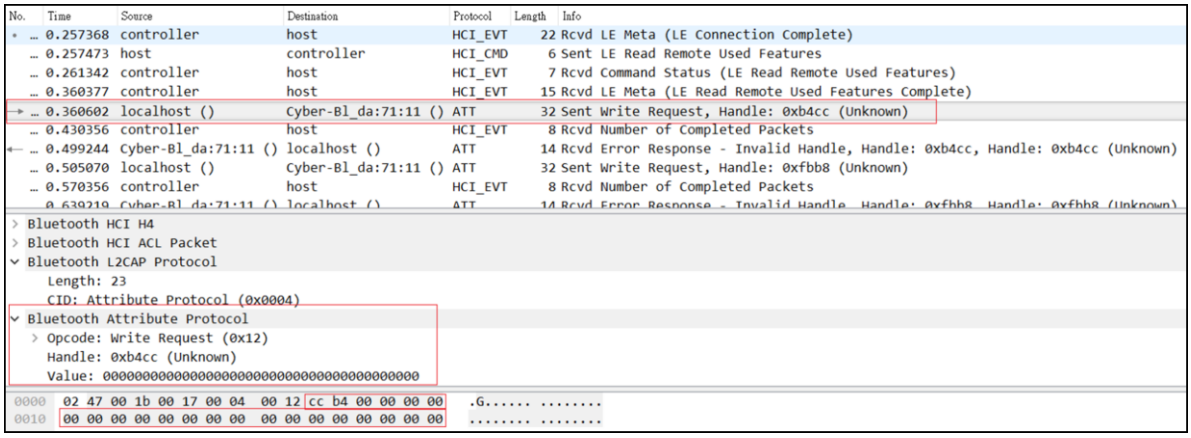


圖八：Advertising Packet 測試

第二種測試則是在連接的情況下，對各個協議層做測試，我們稱為藍芽連線資料測試，由於 Host 和 Controller 各自獨立分開，所以在連線中做模糊測試是困難的，且市面上尚無法找到一款完全開源的藍芽硬體設備，目前僅有 Sniffer 和 Adapter 兩種，這兩種在 Controller 的協議堆疊上是被封裝起來的，所以無法更改其程式碼，來觀察 Controller 封包的流向及內容。第二種測試如圖九所示，我們利用 Bleno 來搭建外圍設備，進而讓我們與外圍設備進行連接，在連接的過程中，找出各個封包的接口，其後讓各個接口能讀入測試資料，並通過原有的傳送流程將測試資料傳送到目標設備。圖十為對 ATT Layer 中的 handle 和 value，共 3 到 20 個 bytes 進行模糊測試的範例。



圖九：藍芽連線資料測試架構示意圖



圖十：ATT Layer 測試

五、結論

由於藍芽通訊技術特性的緣故，對藍芽裝置進行協定的安全模糊測試是困難的，其困難包括跳頻及 Controller 和 Host 的獨立性，本研究提出一低功耗藍芽協定安全模糊測試框架，調查目前可應用於低功耗藍芽協定測試的相關軟硬體，實作出低功耗藍芽協定安全模糊測試平台，並分析其連線的方式，分別進行無線電訊號無連線測試及藍芽連線資料測試。未來我們將利用本研究所提出的框架及實作平台，針對市面上的低功耗藍芽裝置及穿戴裝置進行測試與分析。

參考文獻

[1] “Bluetooth Core Version 4.0 specification,” 2010.
[2] H. Robin, “Bluetooth Low Energy: The Developer's Handbook,” *Prentice Hall*, 2012.
[3] L. Matteo, R. Setola, and J. Lopez, “Cybersecurity of wearable devices: an experimental analysis and a vulnerability assessment method,” *Annual Computer Software and Applications Conference (COMPSAC)*, 2017.
[4] Sławomir Jasek, “Gattacking Bluetooth smart devices”, *BlackHat USA*, 2016.
[5] https://github.com/noble/bleno
[6] M. Ryan, “Bluetooth: With Low Energy Comes Low Security”, Proc. 7th USENIX Conf. Offensive Technologies, *USENIX Association*, 2013.
[7] https://github.com/greatscottgadgets/ubertooth
[8] https://github.com/JiaoXianjun/BTLE
[9] Tommi Mäkilä, Jukka Taimisto and Miia Vuontisjärvi, “Fuzzing Bluetooth Crash-testing bluetooth-enabled devices”, *Codenomicon whitepaper*, 2011.
[10] Apala Ray, Vipin Raj, Manuel Oriol, Aurelien Monot and and Sebastian Obermeier, “Bluetooth Low Energy Devices Security Testing Framework,” *IEEE 11th International Conference on Software Testing, Verification and Validation*, 2018.