Keystroke Logging of a Wireless Keyboard

William Ma, Antony Mbugua, Dickson Poon

Abstract – **This paper investigates the security of Bluetooth. We detail the vulnerability of Bluetooth keyboards to keystroke logging. We demonstrate the feasibility of sniffing Bluetooth transmissions, and give a recorded key pairing, we show that cracking is possible. We also make recommendations for improvement of secure use of such keyboards.**

Keywords – **Bluetooth, Security, Human Interface Device, HID, Keylogging, Sniffing, Privacy.**

I. INTRODUCTION

A. Keystroke Logging

Key stroke logging (or keylogging) is a process of capturing keystrokes from a keyboard. One approach is compromising the kernel of the operating system, providing complete access to keystrokes and also making the logger extremely difficult to detect and counter. Other approaches are to use OS hooks to detect and capture keystrokes, as well as implementation flaws and vulnerabilities in the OS.

A physical keystroke logger can also be created that is attached between the keyboard. It is a small device consisting of an IC and some memory that will intecept the signals from the keyboard, decode and store them and then forward them to the computer over the PS/2 or USB interfaces.

The use of a keylogger allows attackers to gain confidential data such as passwords, which can later be used to bypass other security measures. As such, keystroke loggers pose a serious threat to privacy and security of any computer with a keyboard. The risks posed by software and hardware keyloggers are well known and understood, as they have been in existence for two decades. However the recent growth in wireless technology and devices has created the potential for a new kind of keystroke logging; keylogging over wireless channels.

B. Bluetooth

Bluetooth is a short range communications technology that enables enabled devices to communicate through small ad-hoc networks. Its low power requirements and relatively low cost has made it ideal for use in wireless headsets and keyboards. Bluetooth operates on the non-regulated ISM band: 2.4GHz, where it uses 79 1MHz channels. To avoid interference with other devices, Bluetooth hops frequency at a rate of 3.2K hops/sec or 1.6K hops/sec. The hop sequence is pseudo-random based on the Bluetooth device address and the Clock offset on the master. The core protocols form a four-layer stack consisting of the following elements:

i) Radio – Specifies details of the air interface, including frequency, the use of frequency hopping, modulation scheme, and transmit power.

ii) Baseband (LC)– Concerned with connection establishment within a piconet, addressing, packet format, timing, and power control.

iii) Link manager protocol (LMP) – Responsible for link setup between Bluetooth devices and ongoing link management. This includes security aspects such as authentication and encryption, plus the control and negotiation of baseband packet sizes.

iv) Logical link control and adaptation protocol (L2CAP) – Adapts upper-layer protocols to the baseband layer. L2CAP provides both connectionless and connection-oriented services.

The Piconet is the basic networking unit in Bluetooth. It consists of one Master device to which between 1 and 7 Slave devices can connect. The devices on the piconet synchronize to the master device and share a common frequency hopping scheme. Devices can be in more than one piconet, and goups of piconets that overlap form a scatternet.

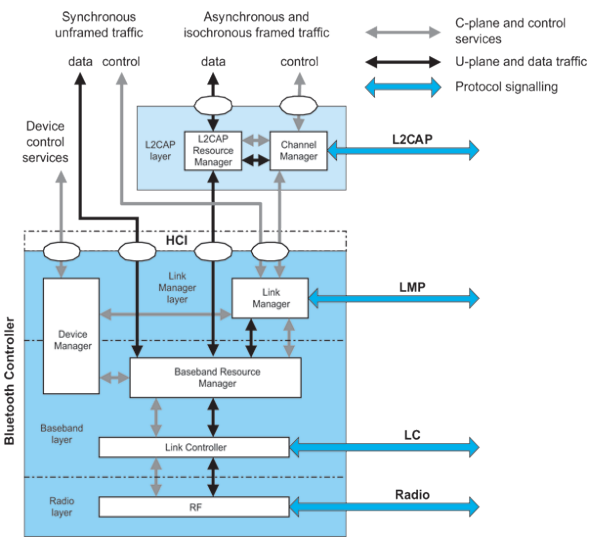

Figure 1: The Bluetooth Architecture [1]

II. BLUETOOTH SECURITY

From its creation, Bluetooth has emphasized security as something of great importance. There are several techniques that for its basis for security.

1) Frequency hopping.
While frequency hopping not only prevents collisions between devices that need to share the same frequencies, but it also makes sniffing a much harder prospect.

2) Discoverable modes
The three discoverable modes address how the device responds to queries

i) Discoverable responds to all inquiries.
ii) Limited discoverable mode makes the device visible for a short period of time.
iii) Non-discoverable mode never replies to an inquiry.

3) Bluetooth Address
The address of a device is a 48 bit field, the first 3 bytes of which correspond to the manufacturer and the last 3 are usually device specific. However sometimes only a limited range of the 3 bytes are used in giving a device a unique address. This is insecure as it allows for brute forcing of addresses to discover 'non-discoverable' devices.

4) Pairing modes also determine if a device can be paired with, or join a piconet. The device can be in pairing or non-pairing mode, meaning it will accept connections or it won't.

5) Bluetooth uses a PIN as a passkey when pairing. This is usually entered on one or both of the devices. It is a UTF8 encoded alphanumeric sequence that ranges between 8 and 128 bits. However, many devices only use a numerical passkey, which makes the PIN much less secure.

6) Security modes
Bluetooth devices operate in several different security modes:

i) Mode 1 has no security or encryption.
ii) Mode 2 has no security until channel is established on L2CAP level (software).
iii) Mode 3 has security initiated before the link setup on the LMP level (hardware).

III. BLUETOOTH VULNERABILITIES

Despite the efforts of the Bluetooth SIG to make the technology secure, there are several significant vulnerabilities. One vulnerability is the ability of an attacker to find non-discoverable devices by passively listening to a preset channel until a device hops by. Then from the Bluetooth preamble in the packet header, the channel access code and other information in the preamble, the remaining 8 bits of the Bluetooth device's address can be brute forced. From this address and the address of the master device, the rest of the hopping sequence can be found.
The two vulnerabilities that we shall focus on in our attempt to log the keystrokes of a Bluetooth HID shall be the weakness of most PINs used for the initial pairing process and the ability to force the two devices to restart the pairing process.

A. PIN attack

This attack is built upon the fundamental weakness of the pairing process, whereby the initial RAND is sent in plaintext over the air. The details of the pairing process are below:
i) Master (A) generates RAND and sends to slave (B). Master and slave generate Kinit = E22(RAND, PIN, PIN_LEN)
ii) A generates RANDA, and B generates RANDB. Random numbers are XORed with Kinit and the result is sent to other device.
iii) Each device creates the link key LKAB by XORing the results of LKA=E21(RANDA, ADDRA) and LKB=E21(RANDB, ADDRB)
iv) Next the two devices authenticate each other by sending an AU_RAND challenge and waiting for the correct SRES. Where SRESB = E1(AU_RANDA, ADDRB, LKAB)

This sequence is shown below:

#	Src	Dst	Data	Length	Notes
1	A	B	IN_RAND	128 bit	plaintext
2	A	B	LK_RANDA	128 bit	XORed with Kinit
3	B	A	LK_RANDB	128 bit	XORed with Kinit
4	A	B	AU_RANDA	128 bit	plaintext
5	B	A	SRESB	32 bit	plaintext
6	B	A	AU_RANDB	128 bit	plaintext
7	A	B	SRESA	32 bit	plaintext

Table 1: The initial Bluetooth pairing process [2]

The first 5 messages are all that is needed to crack the PIN, which is the only part of the authentication process that he/she would not know. This is achieved through a relatively simple process that is detailed in the diagram below. It works by incrementally changing the PIN and calculating the subsequent data and link keys and finally calculating an SRES and comparing it to the sniffed SRESB. If they are the same, then the PIN and Link Key are the same and the attacker can now decrypt all subsequent transmissions.

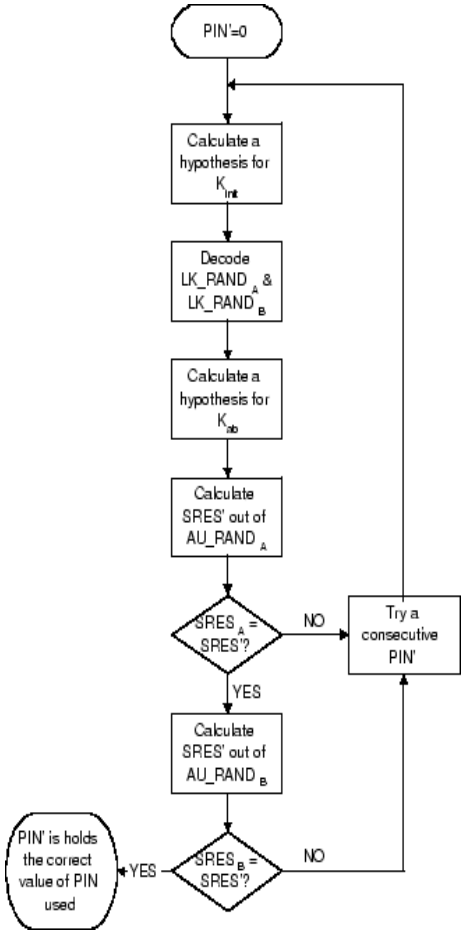

Figure 2: The PIN cracking algorithm [2].

B. Re-Pairing attack

After the two devices have been paired it is practically impossible to break the encryption through brute force. Furthermore, since they have already paired before, they do not go through this step and simple authenticate each other. To do this, the master sends the slave a AU_RAND message and expects the slave to reply with a SRES message. If the slave has forgotten the key then it sends a LMP_not_accepted message. The only recourse is to force the master and slave devices to re-pair. There are several ways of doing this:

i) The first way of achieving this is to inject an LMP_not_accepted message towards the master after it has sent out the AU_RAND. This will convince the master that the slave has forgotten its link key and needs to re-pair.

ii) Alternately, the attacker can send an IN_AUTH message to the slave, convincing it that the master has forgotten the link key and needs to re-pair.

iii) Finally, after the master has sent out an AU_RAND message, the attacker could send out a random SRES reply. This would be the wrong SRES and so after a number of these failed attempts to authenticate, the master would be forced to re-pair.

The re-pairing attack in combination with the PIN attack is sufficient for cracking the messages sent between the master and slave, i.e. To effectively eavesdrop on the communications. It has been demonstrated that cracking a 4 digit PIN can be accomplished in less than 0.1 seconds and a 6 digit PIN can be accomplished in in less than 20 seconds on a modern processor.

IV. KEYLOGGING OF A BLUETOOTH HID

A. The HID Device

Out target device for keylogging is a Bluetooth Human Interface Device. This Rocketfish keyboard is representative of the average Bluetooth keyboard. It uses a HID Bluetooth profile and thus sends out standard HID data over Bluetooth.

Figure 3: The Rocketfish Keyboard.

B. The tools

To sniff the keyboards, Bluetooth transmissions, we used the Bluetooth Analyzer from Frontline Test Equipment, Inc. This provided us with a range of tools to capture and effectively analyze

the packets. To do this we searched for the appropriate packets that corresponded to the initial pair set up process, which were highlighted by their Bluetooth LMP opcode by the Analyzer.

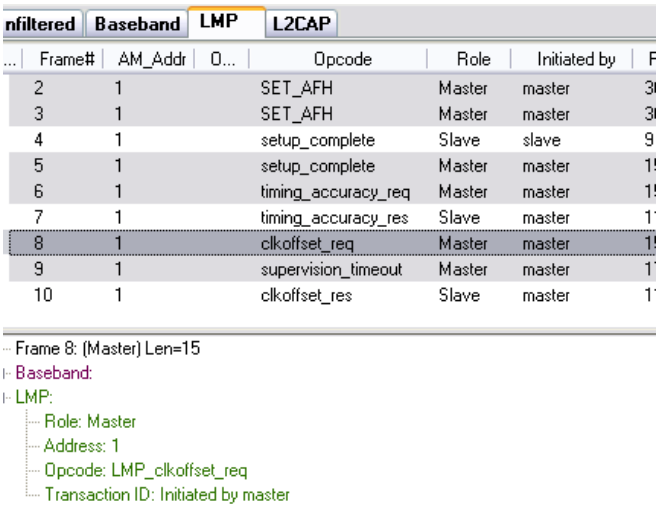

Figure 4: Viewing packets with Frontline BT Analyzer.

After the pairing exchange was exported, we opened it in BTCrack

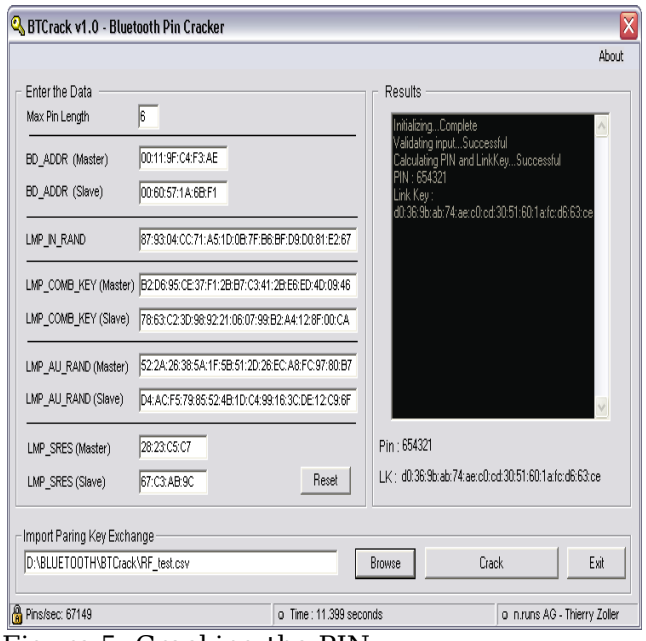

Figure 5: Cracking the PIN.

We were then able to crack the PIN for our keyboard, computer pairing buy simply entering our two BT_ADDRs and the right capture file. The BTCrack software is based off the PIN attack algorithm described above.

While we were able to sniff out and crack the PIN for a paired keyboard and computer, we were unable to find a way to force a re-pairing to occurr. We also realized that our goal of building such a Bluetooth keylogger is not feasible because of the large cost and complexity required to build it.

We did find that the sniffing of Bluetooth packets was possible through a building with a high gain antenna, this greatly increases the threat of spying as an attacker can be in the next building listening in on the transmissions.

V. RECOMMENDATIONS

Now that the keystroke logging potential of Bluetooth keyboards has been demonstrated , it is advisable to not use Bluetooth keyboards for secure or sensitive work. The risk is real and needs to be mitigated by always pairing in a 'secure' location. Avoid using unit (default device) keys and always try to use long alphanumeric passkeys instead of all digits. It is also advisable to be wary of sudden requests for re-pairing as the Link keys are stored in non-volatile memory, there are very few legitimate reasons for such a request.

VI. CONCLUSION

Knowing the addresses of the two devices in question, we were able to sniff the pairing process using the Frontline Blutooth Analyzer suite. We used the collected data packets as the input for BTCrack, whereby we were able to successfully crack the PIN code. It was verified to be the same as the one entered initially during the pairing. We were unable to decode the data within the HID over Bluetooth transmissions due to time constraints. We have successfully shown that it is possible to eavesdrop a Bluetooth keyboard, fortunately it is prohibitively expensive. We were also unable to find a way to successfully force a re-pairing event. This would take specialized and custom built hardware, which we do not have access to nor the resources to create.

REFERENCES

[1] Bluetooth SIG. (2007). How Bluetooth Technology Works [Online]. Available: http://bluetooth.com/Bluetooth/Learn/Works

[2] Shaked, Y. and Wool, A. (2005). Cracking the

Bluetooth PIN [Online].
Available:
http://www.eng.tau.ac.il/~yash/shaked-wool-mobisys05/

[3] Finistere, K. and Zoller, T. (2007). Bluetooth Hacking Revisited [Online].
Available: http://secdev.zoller.lu/

[4] Btdesigner.com (2007). BT Designer Glossary [Online].
Available: http://www.btdesigner.com/ptot.htm

[5] Hager, C.T.; Midkiff, S.F.; “An analysis of Bluetooth security vulnerabilities”. 2003. Wireless Communications and Networking, WCNC 2003.