MITM: Logging a Bluetooth Keyboard

Kimberly Paterson
Cal Poly San Luis Obispo
klpaters@calpoly.edu

Ryan Verdon
Cal Poly San Luis Obispo
rverdon@calpoly.edu

ABSTRACT
TOGO HERE

Categories and Subject Descriptors
H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms
Theory

Keywords
MiTM, Bluetooth Attack

1. INTRODUCTION
Many Bluetooth devices are marketed explicitly for mobile devices such as phones and tablets. Attack against tablets not common

2. BACKGROUND
2.1 Bluetooth
2.2 Bluetooth Security
Frequency hopping: allows for fewer collisions between devices and makes sniffing traffic more difficult, but not impossible since it is known (–find out more about this)

Discoverable modes: limits the vulnerability by limiting the amount of time a device is discoverable by other devices

Bluetooth address (Not sure about this but [**?**] has something to say about it)

PIN authentication no pin mode

Security Modes:

1. Mode 1 no security or encryption
2. Mode 2 no security until channel is established on L2CAP level (software)
3. Mode 3 has security before link setup on the LMP level (hardware)
4. Mode 4 introduced SSP

2.3 Secure Simple Pairing (SSP)
Secure Simple Pairing (SSP) was introduced in revision 2.1 and has remained largely unchanged in the subsequent versions, although some of the authentication protocols have changed [**?**].

SSP has 4 association models:

Numeric Comparison: a 6 digit number is displayed on both devices and a user confirms that the numbers match **Passkey Entry**: designed when one device has input (keyboard) and the other mostly display; the device that has the display shows a 6 digit number, which the user types in on the input device. **Just Works**: one device has no input or display; the user must accept the connection without being able to visualize that the numbers are the same (more in nist) **Out of Band (OOB)**: designed to connect when two devices are touched together, (e.g. Near Field Communication or NFC)

TODO 5 steps of pairing protocol

2.4 MITM Bluetooth Attacks
We focus on man-in-the-middle attacks (MITM) against devices that use Passkey Entry as their primary mode of authentication for pairing. Historically, these attacks have been simple to implement and successful using a variety of approaches.

Among the vulnerabilities NIST lists for Bluetooth versions before 4.0 is the weakness of one-way-only challenge and response authentication used in SSP [**?**]. Since Bluetooth devices often do not have display or input capabilities, Passkey Entry or Just Works modes for SSP are necessary to connect less-functional devices, such as keyboards, to more functional devices such as computers or tablets. When only one device is asking for authentication, it's possible to spoof the keyboard into pairing into Just Works mode and impersonate the keyboard to the victim computer. Despite NIST's statement that the one-way-only challenge affects versions

before 4.0, if an attacker can force a keyboard into Just Works mode, it becomes vulnerable to MITM attacks.

Vulnerabilities: "Device authentication is simple shared-key challenge/response." [?]

2.4.1 Pin Vulnerabilities

NIST recommends that users use a 6-digit random pin when authenticating either in Numeric Comparison or Passkey Entry mode, however, this feature is not mandated. Some operating systems may allow for fewer than 6-digit pins, which are less secure, or let users enter custom pins, which will not be random. Assuming developers follow the recommended protocol, the probability of guessing the pin is one in a million according to NIST [?].

Additionally, NIST claims that since the 6-digit passkey is not incorporated into the link key generation, it isn't useful for an eavesdropper to use. Nonetheless, Lindell showed in 2008 that the pin becomes useful to an attacker when attempting to pair with a device when the passkey is used more than once [?]. Knowing the pin can allow an attacker to connect with the device when the victim attempts to connect his/her devices again. It is possible to force a user to re-pair their devices either by DoS [?] or by making the victims assume that the victim devices forgot the link key[?]. When the devices try to pair again, an attacker can intercept the process and authenticate itself to each device.

If the attacker does not know the pin, he/she can, in fact, crack it as Shaked and Wool showed in 2005 when they were able to crack a 4-digit pin in 0.27 seconds by observing Bluetooth traffic [?]. Similarly, in 2012, Barnickel, Wang, and Meyer were able to able to execute a MITM attack by first jamming the connection between the connected devices, calculate the passkey and then re-pairing with the devices [?]. Both works and Lindell's work exploit the vulnerabilities of the Passkey Entry protocol, where the bit-by-bit evaluation of the passkey by devices can be used to leak the passkey a bit at a time, to determine the passkey quickly. As these works show, computing the pin is alarmingly simple and fairly useful when pins are reused; currently, this is still a feasible attack, but is less likely when pins are not fixed (such as with headsets or other fixed-pin devices).

2.4.2 Key Logging

Regardless of how infeasible it is to guess the password when it is not being reused, some successful approaches target Passkey Entry mode since attackers can force a user to enter the passkey for them. This is espeically useful for devices that rely on Passkey Entry such as Bluetooth keyboards, which do not necessarily have display capabilities and use one-way authentication. Typically, Bluetooth enabled keyboards will connect to a device with display capabilites (e.g. a tablet or PC) and the device will display a passkey that the user must input through the connected keyboard. If an attacker can trick the user into entering the passkey for him/her, sidestepping the pin guessing process becomes quite easy.

The benefit of attacking a keyboard in this way is having direct access to the data being inputed into the keyboard without the need for physical access to the device or the software on either the keyboard or device it's connected to. Additionally, if an attack can successfully execute a MITM attack by logging the keyboard input and then forwarding the data to the victim PC or tablet, the user may not even be aware that their data are compromized.

Keylogging with a MITM attack is theoretically possible and some work has been done toward a seamless attack of this sort. In 2007, Ma, Mbugua, and Poon were able to use a pin attack against a paired computer and keyboard by sniffing the Bluetooth traffic and cracking the pin [?]. However, they concluded that keylogging was infeasible because of the cost. A more successful approach was done by Cuthbert et al. where they first performed a DoS attack to disconnect the PC from the keyboard, and then tricked the keyboard into pairing with their attack PC in unencrypted mode [?]. The attack PC then sends a link request to the victim PC, which connects through the first few stages of authentication and then displays a pin for the user to enter. Thinking that this is a malfunction of the connection, the user unwittingly types the passkey to the attack PC, which then enters the pin into the victim PC. Although Cuthbert et al. were able to connect to both devices, they did not execute a successful keylogger, although they argued it would not have taken much more work.

Our work is based on this last approach, but we implement a successful keylogger that forwards the user's data to the victim device.

3. THREAT MODEL

TODO

4. RELATED WORK

Phan and Mingard show that a malicious device can impersonate a legitimate device and communicate with another legitimate device by making the user enter whichever authentication pin it chooses [?].

BT-Nino-MITM: Bluetooth No Input, No Output MITM forcing devices into Just Works mode [?]

Public key management is a problem [?] theoretical man-in-the-middle attack on the Bluetooth Passkey Entry method [?]

The most relevant research to our own Basic BT attack on keyboard [?]

5. THE ATTACK

There are 4 main steps to executing the attack:

1. Find MAC Address of the keyboard
2. Execute DoS attack against victim
3. Pair with keyboard and victim
4. Forward keystrokes to victim

TODO

5.1 Finding the Keyboard MAC Address

The work relies on the assumption that we can find the MAC address of the keyboard easily and quickly using a Bluetooth sniffer and packet capturing software such as Kismet[link]. Although it is possible to manually program a Bluetooth card, it is possible to obtain hardware that sniffs and integrates with Kismet. Project Ubertooth provides both hardware and a plugin for Kismet that allows you to promiscuously sniff Bluetooth packets and displays the MAC address from whatever device is sending out the packets.

Although we did not implement this approach, we believe that the keyboard's MAC address can be determined within seconds of sniffing, making this step very easy to execute.

5.2 DoS Attack

Pairing with the computer throwing away link keys

Worked for Windows Vista and Linux Ubuntu with BlueZ but did not work for Windows 7 [**?**]

5.3 Getting in the Middle

What makes this attack possible is the unreliability of Bluetooth connections. Users often experience problems connecting devices and understand that disconnections between their devices are common, so service interruptions are not out of the ordinary and are not suspicious. To fix the issue, users will often try to re-pair their device or toggle the connect switch or button to try and connect the devices again.

connect to keyboard in unencrypted mode change Bluetooth address of Bluetooth card to look like keyboard user will enter pin as prompted

5.4 Keylogging and forwarding

NOTES ONE THAT

6. EXECUTING THE ATTACK

To execute this attack, we used completely off-the-shelf hardware and software. The only custom software and hardware would be the incorporation of the Ubertooth device and plugin for Kismet.

6.1 Hardware and Software

For this attack, we wanted to test several versions of Bluetooth as well as different Operating Systems to determine how effective the attack is against different hardware and software. The following are the devices we used for the attack:

Keyboard: we used a basic Bluetooth keyboard that uses Bluetooth V3.0+EDR (Enhanced Data Rate (EDR), an optional speed-up of data transfer)

Attack Device: we used a mid-2012 Apple Macbook Air running OSX 10.8 and Ubuntu 12.10 via VirtualBox with Bluetooh 4.0 card

Victim Devices: we attacked a Lenovo Thinkpad Tablet running Android 3.1 and Bluetooth 2.1+EDR, a Lenovo Thinkpad z61t laptop running Ubuntu 12.04 and Windows 7 with Bluetooth 2.0+EDR, mid-2012 Apple Macbook Air running OSX 10.8

To execute pairing, the Macbook, the Thinkpad Laptop and Tablet all use Passkey Entry authentication where a pin must be entered into the keyboard. Since the attack relies on SSP, which is largely the same accross all versions of Bluetooth since it's implementation in v 2.1, the differences in Bluetooth versions accross the devices should not matter. The largest variable in whether or not the attack is possible is how the OS handles the link keys and sending multiple link requests from a single address.

6.2 Setup

6.3 Attack

7. RESULTS

8. CONCLUSIONS

9. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.