Security Analysis of Bluetooth v2.1 + EDR Pairing Authentication Protocol

John Jersin

Jonathan Wheeler

CS259

Stanford University

March 20, 2008

Version 1

Table of Contents

1 Abstract

Bluetooth is designed for wireless communication between mobile devices. This paper provides a security analysis of the Bluetooth Version 2.1 + EDR pairing authentication protocol. An overview of how we modeled the security properties of Bluetooth using the Murφ verification tool is provided, followed by our findings and analysis. Three different types of attacks were confirmed: rollback attacks, brute force attacks, and a denial of service attack.

2 Protocol Overview

Our focus is on the key generation phase of the Bluetooth protocol. This phase is essentially the session setup phase for a secure exchange of data between two devices.

2.1 Protocol Terminology and Nomenclature

PIN The pin is a secret passcode that is shared between two devices out of band. Normally a user types in the pin into both devices. If one device has a fixed pin, the user types that same pin into the other device.

K_{init} The initialization key is the first key generated by a pair of devices. It is generated from the shared pin, an address, and a nonce. It is used only until a link key is formed, then not used again.

K_{link} The link key is used for authentication at the end of the key generation phase. If encryption is used, the encryption key is derived from the link key.

RAND A random 128-bit nonce.

A The initiating Bluetooth device.

B The responding Bluetooth device.

BD_ADDR The Bluetooth Device Address, used to identify a Bluetooth device.

E A cryptographic hash function. The complex details of how this function is implemented is not a focus of this study, as we assume the underlying cryptography is secure.

2.2 Protocol Outline

1. Pin shared out of band.
2. Bluetooth runs session setup to generate keys.
3. Use keys to communicate privately.

Our focus is on step two with how Bluetooth runs session setup to generate keys. We assume that the pin sharing out of band is secure (step one). Furthermore, we assume that once the keys are established, the cryptography used to communicate with those keys is also secure (step three).

2.3 Protocol Detail[1]

2.3.1 Session Setup, Stage 1

Generate initialization key based on PIN.

First, the two Bluetooth devices share a PIN out of band. Again, we assume that this step of the process is secure, although it's possible that there are some issues with this step mainly involving human error and social engineering factors, such as choosing poor passcodes like "0000", among other errors.

Second, an initialization key based on the inputted PIN is generated by both devices.

Figure 1

$A \rightarrow B$: IN_RAND
$K_{init} = E(BD\_ADDR_B, PIN, IN\_RAND)$

Third, the devices must choose whether they want a combination key or a unit key.

2.3.2 Session Setup, Stage 2

1. If both devices want a combination key, use a combination key:

Figure 2

$A \rightarrow B$: $XOR(K_{init}, LK\_RAND_A)$ $B \rightarrow A$: $XOR(K_{init}, LK\_RAND_B)$
$K_{link} = E(LK\_RAND_A, LK\_RAND_B)$

2. If one device wants a unit key, use a unit key:

Figure 3

A→B: XOR(K_{init}, K_UNIT_A)
$K_{link} = K\_UNIT_A$

Unit key use in Bluetooth v2.1 + EDR is deprecated, but if one device requests the use of a unit key, the other device must rollback to support it. Regardless of the unit key sent, the receiving device must use that unit key.

Unit keys are insecure for a device to use, because if device A connects to B using A's unit key, then later A connects to C using A's unit key, B will be able to listen in. This is avoided by deprecating unit keys, no new Bluetooth devices will request use of a unit key.

2.3.3 Session Setup, Stage 3

Authenticate the two devices based on the established link key.

Figure 4

A→B: $RAND_A$
B→A: E(K_{link}, BD_ADDR_B, $RAND_A$)
B→A: $RAND_B$
A→B: E(K_{link}, BD_ADDR_A, $RAND_B$)

This step will fail if both parties do not have the same link key.

3 Murφ Modeling Details

We chose to model Bluetooth's security properties using the Murφ Verification System, as it's a good tool for doing nondeterministic finite-state analysis on network handshake security protocols in general. Murφ allows one to specify a set of abstract rules for how the protocol behaves at a high level and what sets of conditions must be satisfied at each state. If a given invariant fails to hold in a particular state, and error is reported with the resulting error trace detailing precisely what steps are necessary to end up in that bad state.

As a method of modeling, several natural simplifying assumptions are made. Since we assume that the cryptography itself is secure in this study, no actual encryption takes place in our model within the Murφ verifier. We simply mark a packet as being "encrypted," and all responders/initiators/intruders must respect that flag and treat it as such. Furthermore, our model assumed that the intruder has full control of the network. That is, he may intercept, record, block, and generate packets at will. This assumption allows one to find potential man-in-the-middle attacks.

4 Analysis of Found Attacks

Our analysis of the Bluetooth v2.1 + EDR Pairing Authentication Protocol confirmed three different types of attack: rollback attacks, brute force attacks, and a denial of service attack.

4.1 Rollback Attacks

Two variations of rollback attacks were found. These are the most serious attacks found.

4.1.1 Attack I

The first rollback attack found involves a man-in-the-middle attacker. Both honest devices request use of a combination key, but both receive requests (apparently from their honest partner) to use a unit key.

Both parties want a Combination key:

Figure 5

$A \rightarrow I$: $XOR(K_{init}, LK\_RAND_A)$
$B \rightarrow I$: $XOR(K_{init}, LK\_RAND_B)$

But both get a unit key in response:

Figure 6

$I \rightarrow A$: $XOR(K_{init}, K\_UNIT_I)$
$I \rightarrow B$: $XOR(K_{init}, K\_UNIT_I)$
$K_{link} = K\_UNIT_I$

This forces both parties to rollback to support the received unit key request, even though both actually sent combination key request to one another.

4.1.2 Attack II

The second rollback attack builds on the first type of rollback attack. After setup completes the first time, and the first rollback attack has taken place, the honest parties periodically refresh their link key. The key refresh is done by running the session setup again starting with the link key messages. The intruder blocks these messages and sends messages which force the devices to keep the same key.

Intruder sends back all zeros:

Figure 7

I→A: XOR(K_{link}, K_{link}) = 0000…
I→B: XOR(K_{link}, K_{link}) = 0000…
$K_{link} = K_{link}$

This is possible because the request is simply a XOR of the old key and the new key. So if the XOR is all zeros, then the old key and the new key will be the same key.

4.1.3 Fix for Rollback Attacks

1. Unit keys are deprecated (starting with Bluetooth v1.2), so it would be best not to maintain backward compatibility. If a unit key request is received, terminate the connection. All modern devices should support combination keys, minimizing the impact.

2. Bluetooth should not chain the new link key with the old one. Instead, the entire session setup should be rerun starting with the first nonce that is used to generate a new initialization key. This guarantees that the new link key will be fresh and will not depend on the old link key.

The rollback attacks stem from an attempt to maintain backward compatibility. Our attacks show that backward compatibility come at a cost, however, by implementing our second fix, backward compatibility could have been achieved without our more serious version of the attack.

4.2 Brute Force Attacks

Two variations of brute force attacks were found. The question here is: Is it possible for the intruder to obtain a plaintext-ciphertext pair, which can then be used to computationally determine the PIN used?

4.2.1 Attack I

The first brute force attack involves an intruder listening to two devices pairing. This is possible because a RAND nonce is sent in the clear and sent back encrypted as shown in Figure 4. This version of the attack is only possible when two devices are initiating a conversation, and the attacker records the messages.

4.2.2 Attack II

Another very similar brute force attack can be done without a pairing of two devices occurring. In this version, an attacker can collect enough data from a device to brute force the PIN without any special circumstances.

First, an intruder device initiates a connection with a responder by sending a RAND nonce. The responder will use this nonce and the PIN to generate an initialization key. The attacker can then send a unit key request as shown in Figure 3 (Session Setup, Stage 2). Finally, the attacker can then send another RAND nonce. The responder will then send back the encrypted version of that nonce as shown in Figure 4 (Session Setup, Stage 3). It is important to note that the attacker does not know how the responder will interpret the unit key request, as the attacker does not yet know the initialization key with which it was XORed.

The attacker can make repeated guesses as to what the PIN may be and attempt to decrypt the conversation on that premise. If a guess of the PIN results in a conversation for which the encrypted authentication nonce matches the one send in the clear, then the PIN is saved by the attacker. PIN's can be as small as 1 byte, making this attack potentially very dangerous, however, it is only possible because a plaintext ciphertext pair is available.

4.2.3 Fix for Brute Force Attacks

The Bluetooth v2.1 + EDR specification states that if an initiating device fails at the authentication step (Session Setup, Stage 3), then the responder device must impose an exponentially increasing delay on the initiator being allowed to attempt to establish a new connection. This effectively blocks online brute force attacks. However, offline attacks, as above, are not only possible, but also entirely more powerful than any online attack.

To prevent offline attacks Bluetooth should prevent an attacker from ever seeing a plaintext ciphertext pair. This can be done easily by encrypting the authentication nonces both ways in Figure 4. The responder would then reply with a message such as the nonce plus one. Using this method the attacker cannot verify guesses of the PIN entirely offline; guesses must be verified online, leveraging the built in online exponential backoff defense.

The authentication nonces are likely sent in the clear to avoid unnecessary processing cost. However, because PIN's can be as small as 8 bits, it is important to avoid these attacks, as once a PT/CT pair is available, computation of the key becomes trivial, and online attack defenses have no effect.

4.3 Denial of Service Attack

If an attacker sends a series of nonces to a responder, the attacker can force the responder to do 3 cryptography operations for each 2 nonces that it sends, as depicted in Figure 1 and Figure 4. If the exponential back off feature of the responder denies further requests from the attacking device, then the attacking device can simply spoof its originating device address and make more new initiation requests.

Furthermore, the Bluetooth v2.1 + EDR specification states that while this Bluetooth processes are executing no preemption may occur. Therefore, no real work can be done while this computation is being performed. This fact allows an attacker to perform a DoS attack on a solo device and stop the device from functioning as normal (e.g. a phone could not make phone calls).

4.3.1 Fix for Denial of Service Attack

Some possible fixes for this attack include:

1. Perform authentication after K_{init} is formed between Session Setup, Stage 1 and Session Setup, Stage 2. This is a one-time cost for the lifetime of the two devices, as the initialization key will always be the same between them. This improves the ration of work done in the protocol.

2. Use preemption. Preemption is not allowed in the Bluetooth specification in order to avoid reflection attacks (where a party reflects an authentication nonce back it its sender to get the correct response). However, other sufficient security measures already exist in the Bluetooth specification to prevent reflection attacks. Using preemption shouldn't be a security risk.

This attack is dangerous largely because of the preemption issue. This represents a tradeoff between defense against two attacks, a reflection attack and a DoS attack. Because the reflection attack is not possible anyway, we believe that the protocol should have balanced its defenses by mitigating the DoS attack.

5 Conclusion

Our work uncovers new vulnerabilities in the Bluetooth protocol. None of these directly compromise the basic security properties of the protocol such as secrecy or authentication. However, the attacks and their corresponding fixes are instructive. The Bluetooth protocol makes repeated tradeoffs between security and other issues such as hardware requirements and backward compatibility. The vulnerabilities shown in this report are all the result of such tradeoffs. By discovering these vulnerabilities and analyzing the tradeoffs that led to them, we feel we have shed light on security in the real world.

6 References

[1] Bluetooth Core Specification Version 2.1 + EDR. 27 July 2007. Bluetooth SIG, Inc. 20 March 2008 <http://www.bluetooth.com/Bluetooth/Technology/Building/Specifications/Default.htm>.