CODENOMICON defensics™

Codenomicon whitepaper:

Fuzzing Bluetooth

Crash-testing bluetooth-enabled devices

- Tommi Mäkilä, Jukka Taimisto and Miia Vuontisjärvi -

1 Introduction
2 Misconceptions about Bluetooth security
3 Broken input and fuzz testing
4 Test setup
5 Test execution
6 Results
7 Conclusion

CODENOMICON Ltd. | info@codenomicon.com | www.codenomicon.com

Tutkijantie 4E | FIN-90590 OULU | FINLAND | +358 424 7431
12930 Saratoga Avenue, Suite B-1 | Saratoga, CA 95070 | UNITED STATES | +1 408-414-7650

1 Introduction

Bluetooth technology is used in many different devices: computers, mobile phones, handsfree equipment, and the car audio systems for example. When the applications become more critical, the importance of security and robustness testing is highlighted.

Lately, more attention has been paid to security of Bluetooth systems, but the focus has been on pairing and authentication. Handling of malformed data has been largely ignored. Yet it is the malformed data, broken inputs that Bluetooth systems have little tolerance for. Test results from plugfest events are worrying: failure rate of over 80% is devastating.

Malformed input may cause Bluetooth device operation to slow down, or device may show unusual behavior or crash completely. This causes degraded quality of service and even denial of service (DoS). In a worst case scenario, malformed input can be used by an outside attacker to gain unauthorized access to Bluetooth device.

2 Misconceptions about Bluetooth security

Bluetooth is usually not perceived as a security or quality threat neither by equipment manufacturers nor by consumers, although the problems and risks are very real. There are several common misconceptions about Bluetooth security that undermine the need for Bluetooth testing. These misconceptions are listed and contradicted below.

Pairing/authentication protects us

To protect the private data on Bluetooth device, a process called pairing is used. Two devices need to be paired to communicate with each other. A device receives a connection request from another device with which it is not yet paired, and the user then accepts the request to pair the devices. Often the choice is confirmed with a pin-code authentication.

As useful as pairing and authentication is, it is not enough to protect the Bluetooth system. First of all, pairing always leaves the lowest layer of Bluetooth stack, core protocol L2CAP and Service Discovery Protocol SDP exposed. This is because SDP is used to announce available services on devices and for usability it needs to be accessible without pairing, and SDP in turn runs on top of L2CAP.

What is more, in Bluetooth 2.0 and older, the pairing is usually verified with a 4 digit pin code. As people in general tend to go for very easy pincodes, the code is typically either 0000 or 1234. Also many devices such as handsfree units have a hardcoded pincode, usually 0000, which cannot be changed. So instead of trying to crack very sophisticated algorithms, the pairing can be "circumvented" by concentrating on breaking just 4 digits.

It is not just easy pincodes, pairing in general leaves a lot of responsibility to the user. Therefore, social engineering can be efficiently used to get pass the pairing for example by using familiar/deceptive device names to initiate pairing.

Pairing offers traditional security only, in the form of authentication and encryption. It does not provide protection against protocol level robustness shortcomings (one might in purpose

pair with a badly implemented device, which ends up sending a malformed packet and causes a crash).

Pairing functionality it itself is vulnerable to errors caused simply by mis-implementation or inability to handle invalid data. For example, Secure Simple Pairing in Bluetooth 2.1 offers different pairing modes, one being JustWorks. JustWorks is used when the target device does not have any means to display verification codes or allow meaningful user input. This mode can sometimes be abused by using it against more capable devices that accept the JustWorks pairing mode even though they should use the more secure numeric comparison method.

Worst of all, some devices have suddenly stopped requiring pairing after they have crashed during robustness testing or fuzzing. This effectively means that the attacker can first target his attacks on an underlying security feature, disable that and get full access to the device and more feature-rich application protocols.

Conformance testing is covered and that is enough

The standard Bluetooth conformance test practices, such as IOP testing at UnPlugFests and running the Profile Tuning Suite (PTS), help devices to gain a certain baseline for reliable communication between devices in the real world. It also efficiently eradicates situations where for example misinterpreted specifications hinder devices from communicating properly.

However, in majority of the cases these errors still occur in the valid scope of the specifications. A device using incorrect settings for a specific operation, for example, may cause problem situations. Even in-depth conformance testing does not prepare the device to handle robustness shortcomings from unexpected broken inputs, where the errors lie in the handling of structures and message sequences.

There is nothing worth attacking, and exploitation is only theoretical

This assumption usually spawns from the fact that Bluetooth is used in mobile devices, that are usually limited to being used by a single person (as compared to for example routers which serve multiple users simultaneously).

However, these devices often house very sensitive personal data and the number of devices in active use is nothing short of astronomical: almost everyone owns a mobile phone, many modern cars have a Bluetooth enabled edutainment systems, many people use Bluetooth enabled handsfree equipment for phone calls and so on. As a result, any discovered vulnerability can potentially be used against a very vast number of devices.

Personal devices such as mobile phones are used to access a vast number of more critical services such as email, voice mail and web services. All the keys and passwords to those services are typically stored in the mobile phones.

Since available Bluetooth stacks are relatively limited when comparing to the number of different products, any vulnerabilities discovered within a specific stack can easily be used against a large number of different devices. The stacks used in various devices is usually either already known, or the information is easily obtainable.

The fact is, exploitation is not theoretical: there are Bluetooth exploits available and they are being used to target Bluetooth devices, whether just to bother people with denial of service or actually gain access to Bluetooth device. Not to mention that not all malformed input comes from a purposeful attack, but may be caused accidentally by a malfunctioning device.

3 Broken input and fuzz testing

When a Bluetooth device receives an invalid message, it is likely to give an abnormal response: the device may crash, for example, or it may stop requiring pairing process, or it may allow installing and running malware. These invalid messages, that contain broken input may come from connecting to a non-conforming device, or from an outside attacker.

The abnormal reactions are due to vulnerabilities in software, mistakes in software. To harden the system against invalid input, these vulnerabilities need to be found and fixed.

Fuzz testing is a security testing method that uses broken input to find vulnerabilities. In a controlled surrounding, invalid messages are fed to the system under test in purpose. The system's behavior is then carefully monitored to detect abnormal responses. When something unexpected happens, it indicates that there is potentially exploitable vulnerability in the software that, once discovered, can be fixed to eliminate the problem.

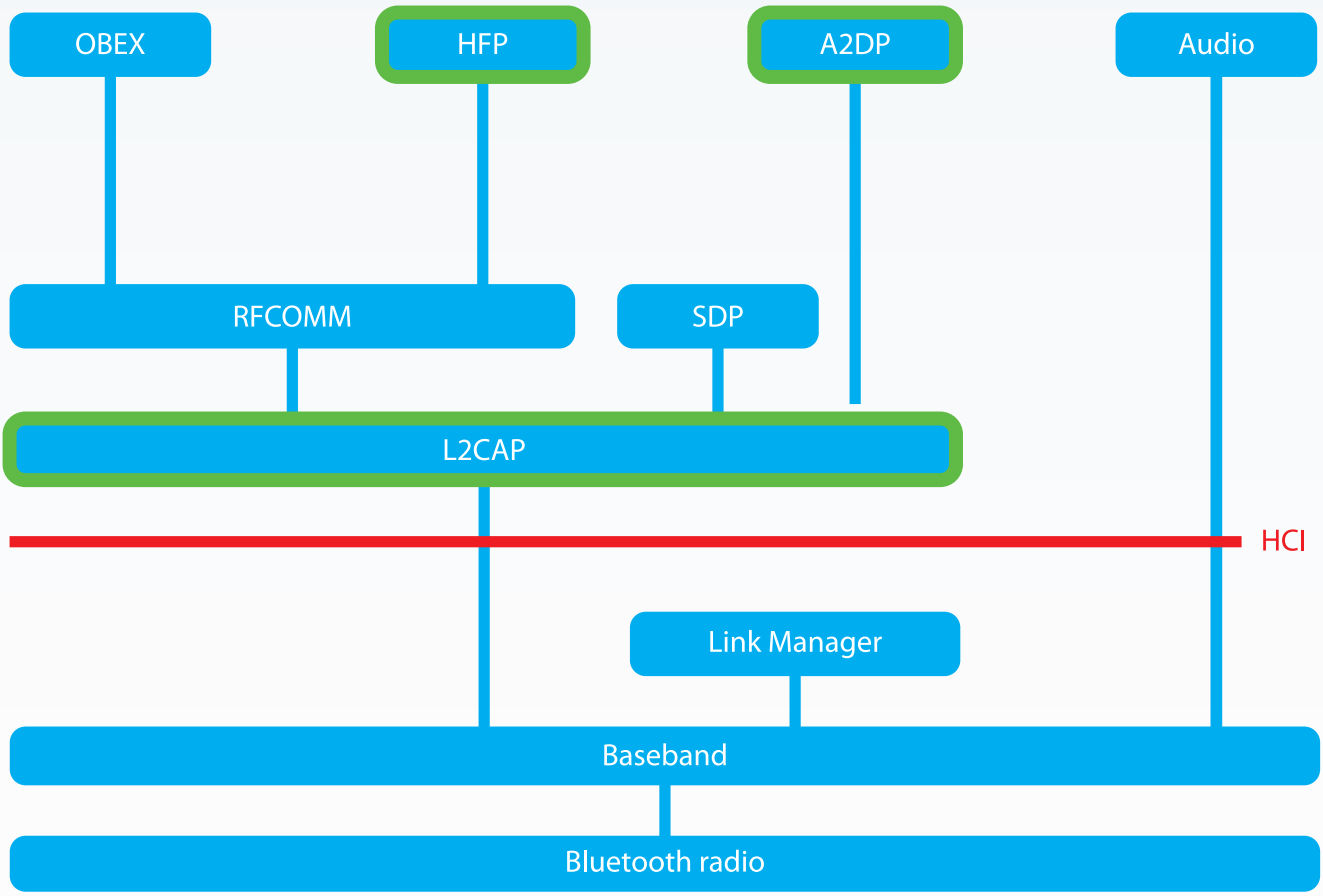
The tests described in this paper were conducted using a fuzz test tool designed for testing Bluetooth systems.

4 Test setup

Bluetooth stack

Bluetooth is a layer protocol architecture that consists of core protocols, cable replacement protocols, telephony control protocols, and adopted protocols. Mandatory protocols for all Bluetooth stacks are LMP, L2CAP and SDP. Additionally, HCI and RFCOMM protocols are almost universally supported.

The following figure illustrates a simple version of a Bluetooth protocol stack. This whitepaper will focus on protocols highlighted in green.



Tested profiles and protocols

Bluetooth device has to be able to interpret Bluetooth profiles. They specify general behavior that Bluetooth enabled devices use to communicate with each other. There are a wide range of Bluetooth profiles that describe many different types of applications or use cases for devices.

For this paper, we concentrated on testing in-car Bluetooth carkits, after-market car stereo systems with Bluetooth support and Bluetooth headsets. Bluetooth profiles used by these systems are usually HFP, that provides the ability to conduct phone calls with mobile phones, and A2DP, that provides the ability to play music from music collection located on a mobile device.

In addition to testing the supported profiles, tests were conducted also for the underlying Bluetooth core protocol, L2CAP. This core protocol is used to transmit the profile payloads and thus is also affected. Also, L2CAP protocol layer is usually exposed, even if the access to other profiles would require security mechanisms, since SDP transactions need to be allowed before any pairing procedures.

The scope of the testing was limited to actual protocol messages as described in corresponding specification. For HFP testing, this means that the test material consisted of anomalies for different AT commands. AT commands are normally used for a standard Service Level Connection establishment procedure between an Audio Gateway and an Unit device. For A2DP, the test material consisted of anomalies for basic AVDTP (Advanced Audio Distribution Transport Protocol) messages. AVDTP messages are normally used for various operations such as discovery, parameter negotiation, stream establishment, audio controls between an A2DP Audio Sink and an A2DP Audio Source devices. In both HFP and A2DP tests the actual audio transfer was either largely left unmodified or omitted altogether. This is because different audio formats and streaming are not in the scope of the referred protocol specifications.

DUT setup

Device under test (DUT) setup depends on the profile or protocol to be tested. When testing L2CAP protocol, it is usually enough just to turn Bluetooth on. This is true in particular with after-market car stereos. To ensure that DUT is ready to accept incoming connections, it should be put into discoverable or pairing mode. In this mode the DUT expects and accepts incoming connections from all devices at least to SDP PSM.

When testing the upper level profiles, the DUT setup normally involves pairing procedure. Once the tester has been paired with DUT, the DUT will allow the tester to connect using the profiles.

Sometimes the DUT setup is not so straightforward. We have observed some difficulties, especially with in-car Bluetooth systems. These systems might check (using SDP) the supported profiles from tester as a part of pairing procedure and refuse to establish the trust relationship if the tester does not respond to the SDP query with right profile information. Sometimes it can be quite difficult to know exactly what the DUT is expecting. We have equipped our tester with SDP server which contains profile information which should be accepted by most devices.

Some (especially in-car) systems may actually contain two distinct Bluetooth stack implementations. One for phone call related profiles and one for audio playback. In this case the tester needs to be paired with both stack instances separately. Also, don't forget to test the core protocols on both stacks since the stacks may very well be completely different from each other.

5 Test execution

Testing procedures

L2CAP: L2CAP connection from the tester to DUT is initiated. Normally, the connection is made into SDP PSM because SDP should be supported by all Bluetooth devices.

A2DP: Pairing procedure is performed prior to testing. Both an AVCTP (Bluetooth Remote Control on PSM 23) and AVDTP (Bluetooth Audio Streaming on PSM 25) connections to the DUT are opened. Audio stream parameters are then configured, and media channel for stream delivery is opened.

HFP: RFCOMM connection is opened using a specific channel. Once the RFCOMM connection is established, a Handsfree Service Level connection sequence is performed. The role of the device (Audio Gateway or Unit) dictates whether the HFP AG or HFP Unit test suite is used to test the DUT.

6 Results

During the Bluetooth carkit bonanza, we tested 15 car kits, 5 mobile phones, 3 headsets and a BT picture frame. Out of the 24 test targets, two had such unusual operating logic that we could not get any test results. All the other failed at one point or another.

When testing embedded devices, anything can happen when software module malfunctions. We have seen that programming error in Bluetooth module causes the complete in-car infotainment system to crash.

When testing hands-free units, some units became completely unresponsive as a result of a programming error. Removing the battery or letting the unit run out of power can sometimes fix the problem, but if that does not help, then the unit needs to be reprogrammed.

It is recommended that DUT is completely reflashed after the tests have been concluded. It is not a rare occurrence that devices memory contains invalid pieces of data or that the device acts abnormally after testing has been concluded. Two of the tested car kits had to be taken back to the dealer for reflashing due to abnormal behavior after they seemingly recovered.

To get as reliable results as possible, the testing was conducted on development environment where proper monitoring of the DUT is possible.

See the tables below for more information.

Headsets

Test target	Test result	Notes	Crashed with
Headset1	Fail	Died permanently	HFP
Headset2	Fail		L2CAP, HFP
Headset3	Fail	Died permanently	HFP

Mobile phones

Test target	Operating system	Test result	Notes
Phone1	Android	Fail	Worked ok after boot
Phone2	Win7	Fail	Worked ok after boot
Phone3	Android	Fail	Worked ok after boot
Phone4	Proprietary	Fail	
Phone5	Android	Fail	

Carkits

Test target	Test result	Notes	Crashed with
Carkit1	Fail	started the next day	L2CAP, A2DP
Carkit2	Fail	After-market carkit, worked ok after boot	A2DP
Carkit3	Fail	After-market carkit, worked ok after boot	L2CAP
Carkit4	Fail	After-market carkit, worked ok after boot	L2CAP, A2DP
Carkit5	Fail	Worked ok after boot	L2CAP
Carkit6	Fail	After-market carkit, worked ok after boot	HFP, L2CAP
Carkit7	N/A	Unable to test due to strange operating operating logic	
Carkit8	N/A	Unable to test due to strange operating operating logic	

Test target	Test result	Notes	Crashed with
Carkit9	Fail	After-market carkit, worked ok after boot	L2CAP, A2DP
Carkit10	Fail		L2CAP
Carkit11	Fail		L2CAP
Carkit12	Fail		L2CAP
Carkit13	Fail	After-market carkit, worked ok after boot	HFP, A2DP
Carkit14	Fail	Car manufacturer's standard carkit	L2CAP
Carkit15	Fail		A2DP

Other

Test target	Test result	Notes	Crashed with
BT Picture Frame	Fail	Worked ok after boot	L2CAP

7 Conclusion

Security and robustness of Bluetooth equipment currently in the market is poor, perhaps even worse than anyone expects. When tested with intelligent fuzz testing tools designed for Bluetooth testing, every test target failed.

The most worrying finding was the unreliable behavior of L2CAP layer. While a few years back there seemed to be light at the end of the tunnel as the failures were moving up the Bluetooth stack, L2CAP robustness showed some improvement. Only for a moment though, as recent tests again show a steady decline in results. Now, most of the equipment tested crashed within the first 100 cases of L2CAP protocol tests. This is a problem in particular since L2CAP does not require pairing. This means, that L2CAP can be targeted without the user accepting or even noticing the connection.

One can only guess, why the manufacturers choose not to improve the security and robustness of Bluetooth devices. Misconceptions described earlier are one thing to undermine the need for Bluetooth security testing. Also, Bluetooth equipment are consumer products with one individual user at a time, which also reduces the perceived importance of testing Bluetooth systems. However, the number of individual users is huge, and applications are becoming more and more critical. Therefore, Bluetooth security and robustness certainly should receive more attention than it is currently getting. After all, tools and methods to improve performance and reliability already exist.