The art of *Bluedentistry*:

Current security and privacy issues with Bluetooth devices

Leith Caldwell, Stefan Ekerfelt, Armin Hornung and Jean Yanni Wu

Computer Science & Engineering
University of Washington
{leith, ekerfelt, hornung, jeaneis}@cs.washington.edu

December 13, 2006

Abstract

As more personal, mobile electronic devices are equipped with the Bluetooth technology for short-range wireless, users of these devices often unknowingly have them enabled to be discoverable all the time. This can lead to security threats as well as privacy concerns, however most people are not aware of these risks or do not perceive them as issues. In this paper, we highlight some of the vulnerabilities of the Bluetooth protocol and extend the mobile Denial of Service attack to Bluetooth. We have found that currently popular Bluetooth-enabled mobile devices are susceptible to DoS attack, since these devices display pop-up notifications whenever they receive an incoming request. Tracing the exact position of the devices is more difficult due to the high range of Bluetooth sensors, although it is still possible using signal strength.

1 Introduction

The Bluetooth technology for short-range wireless is becoming more and more ubiquitous. Most mobile phones are equipped with it for data exchange or connection to peripherals like headsets, and the number of laptops equipped with Bluetooth by factory default is growing. However, possible security and privacy issues concerning Bluetooth and mobile devices have not been explored until very recently, resulting in an increasing number of exploits being published. Even though basic security measures like invisibility, authentication and encryption exist, they are rarely used effectively.

Our intention is to study what kind of threats to security and privacy exist today: How much information can be obtained over the public Bluetooth communication channel, and how can it be used by an eavesdropping adversary? For example, is it possible to track people via Bluetooth on their mobile devices? How prevalent are security risks in current mobile phones?

We mainly focus our study on mobile phones and their connection to other devices over Bluetooth. The rest of the paper is outlined as follows:

In Section 2, we will give some general information on Bluetooth, how it works, and some of the possible issues related to security and privacy. A brief overview of related work follows in Section 3. Then in Section 4, we will describe how we approached our set goals to study the current state of privacy and security. In Section 5 we provide some analysis of our traces and vulnerability testing followed by discussion in Section 6 and our conclusions in Section 7.

2 Background

Bluetooth [1] is a short-range wireless protocol with a typical range of about 10 meters[1], operating at 2.4 GHz just like Wi-Fi LAN. The Bluetooth protocol runs on several layers. The lowest layer is L2CAP, building the security layer of Bluetooth, followed by RFCOMM which emulates serial connectivity profiles over Bluetooth. On top of that, OBEX (Object Exchange) is executed to provide file transfer services.

Bluetooth allows devices to restrict connections only to previously *"paired"* devices, or even to make a device invisible to any unknown device. In practice,

[1] With special equipment, the range can be extended to more than 1 km.

this is hardly used, just like the possibility to encrypt Bluetooth traffic. Often, the default setting is to be always visible in the Bluetooth neighborhood.

Like network devices, each Bluetooth device or dongle has a unique MAC address assigned to it. There are tools to spoof the MAC on certain devices [11], but usually one can assume that it is unique, especially for mobile phones. The manufacturer of a device can be inferred by the first 24 bit of the MAC address, the Organizationally Unique Identifier (OUI)[14]. The following 24 bits are usually randomly assigned to the devices, but by the way a phone or other device responds to a *Service Discovery Protocol* (SDP) inquiry, its model and sometimes even firmware version can be identified. This service fingerprint hash, or Blueprint [4] can be obtained for all popular phones and stored in a database, together with possible vulnerabilities.

Further information about the device and its owner can be inferred from the name of a device. By default, this is often the model of the phone, but people also tend to change it to their own name.

2.1 Security Issues

There are a number of possible attacks against the different layers of Bluetooth, some generic and some device-specific, exploiting faulty or insufficient implementations. Since the Blueprint enables the identification of a phone's model, an adversary can manually scan for all devices in range and then launch model-specific attacks. Alternatively, a Bluetooth worm could spread itself by targeting different devices with specially tailored attacks. Some common attacks are:

- **Bluesnarfing:** Without alerting the owner, an adversary can connect to a mobile phone, retrieving personal data such as phone book or calendar files. The connection is done via OBEX, e.g. directly on the file containing the phone book.
  A more advanced attack *Bluesnarf++* opens an OBEX ftp connection to the target with full interactive access to the file system.

- **Bluebug:** Again secretly, one can establish an AT connection to certain phones to control them remotely. This can include altering and reading phone content (address book, calendar, SMS inbox) or remotely controlling the phone (initiate calls, send SMS). So far, this vulnerability has only been reported on older Nokia phones.

- **Denial-of-Service:** The phone receives a number of repeated connection requests, making it impossible to use. In particular, it becomes nearly impossible to turn off Bluetooth to evade the attack. More targeted attacks send specially crafted packets which force the Bluetooth stack on the phone to crash [12].

- **Bluejacking**: Recently, it has become quite popular to communicate over side channels of the Bluetooth protocol. For example, some phones directly show the name in a vCard[2] upon reception, so people make use of that by sending messages in vCards. The same can be done using the pairing process, when the targeted device shows the complete name of the requesting phone. When done intentionally as a conversation, this sometimes is also referred to as *Bluechat*.
  We don't consider this a critical attack, no matter how annoying this anonymous side-channel communication may be.

- **Cracking PIN and pairing process**: If security measures are in effect, they rely on trusted devices being paired with the own device, before granting access to any critical ressource like the phone book. In some cases, it is possible to sniff and decode the traffic of the pairing process, with that retrieving the PIN. This, used together with MAC spoofing, enables and adversary to pretend to be a trusted, paired device. We will not cover those kinds of attacks.

2.2 Privacy Issues

Without exploiting any security holes, just using the publicly available data which Bluetooth uses to communicate is sufficient to threaten the privacy of the owner of a device. With the MAC address it is possible to uniquely identify each device, distinguishing the manufacturer and even the device model by the Blueprint. Together with the relatively short range of usually 10 meters, detailed movement and habit patterns can be created. This all relies on users leaving on Bluetooth functionality on the mobile devices they carry around each day. Recent research in 2005 and 2006 has shown that this assumption is valid and detailed tracking is indeed possible. [2, 3]

Even if Bluetooth visibility is turned off, it is possible to probe for devices in range, because every device responds to an inquiry directed at its MAC, such as a simple `l2ping` in Linux. To avoid scanning through

[2] Standard defining the format of an electronic business card

all 2^{48} possible addresses, the scanning can be limited to certain ranges of manufacturers as given by the OUI [15], or to only previously seen devices. With the latter approach, the list of devices to probe stays short enough. Even though constantly hidden devices stay undetected, at least spontaneously visible ones, e.g. to do pairing or data transfer, can not hide afterwards. More speedup can be gained by using a number of Bluetooth dongles on one PC simultaneously.

To monitor movement patterns, a network of sensors can be deployed either choke points or covering large areas with multiple sensors, using triangulation for detailed location.

3 Related Work

[2, 3] were, to our knowledge, the first large-scale studies conducted on tracking people via Bluetooth devices. They both monitored a university environment, with the first study also extended to the CeBIT 2004. The social aspect of Bluetooth tracking, behaviour and device naming is covered by [8]. These sudies take a very similar approach to ours.

Blueprinting was introduced by Herfurt and Collin [4] in 2004, but our study uses this technique to identify single phone models in relation to tracking a large number of targets.

Bluetooth worm infections were investigated in [5], with the feasibility shown and the spreading simulated. While we do not explore this explicitly, it is raised as a concern that complements our work.

4 Methodology

4.1 Monitoring Script

For monitoring the Bluetooth traces and device details, a monitoring script was deployed on the sensors [3]. For simplicity, we used *braces* from the Shmoo Group as a starting point, written in Perl and first used at *Blackhat US 04* [6]. It utilizes tools of the standard Linux Bluetooth stack BlueZ [10] for scanning: `hcitool` and `sdptool`. The main modifications to the script were:

- To run in an offline environment, the collected data was stored in local files and thus a timestamp has to be recorded on scanning time.
- Additional details for each seen device were acquired, besides the MAC-address and clear name: Bluetooth class and Bluetooth fingerprint (see next item). After inital testing, the script was revised for scanning speed and space efficiency.

  The details were then recorded only once per device, under the assumption that the given clear name was not changed over our monitoring time of about at week. That reduced the size of the traces to the order of 1 MB and sped up the scanning process.
- To get unique identifiers for each phone model, we recorded a fingerprint based on the listing of services each phone has to offer over SDP, called a Blueprint [4]. This was also used to observe some simple mobile model statistics.
- To conduct this privacy study, we had to follow university regulations concerning human subjects and privacy. Thus, we limited our complete data collection to MAC addresses known to us, belonging to students from our class in Computer Security. This means that this part of our research can only be seen as a feasibility study, preparing eventual future long-term research.

The separate lists of devices per sensor leads to possible duplicates coming from different sensors, however these can be resolved in offline processing. The details of a device are only stored if no data is missing. No details were stored if the target is out of range when doing the detailed scan, it didn't report a proper name, class or SDP listing, or a timeout occurred. That caused the monitoring script to probe that device the next time it became visible.

4.2 Infrastructure

A total of four sensors were dedicated for Bluetooth scanning and were deployed at various locations in the Paul Allen Center of University of Washington. Since our scanning scripts are implemented with Perl using the BlueZ protocol stack, each system was running a Linux distribution with the BlueZ kernel modules and Perl installed. To make the system deployment and installation as smooth as possible, we decided to run the scripts on the PCs under a Linux Live CD. Due to lack of reliable network connectivity, each system stored the scanning data on a local disk, which later was manually collected and inserted into a central database running PostgreSQL. In order to not erroneously skew the

[3] available upon request

System	Linux Distribution	USB Device	RAM	Processor
Desktop #1	Linux Knoppix 2.6.17	USB D-Link DBT-120	512 MB	Pentium 4 CPU 2.80GHz
Desktop #2	Linux Knoppix 2.6.17	USB D-Link DBT-120	512 MB	2 * Pentium 4 CPU 3.00GHz
Notebook	Linux Knoppix 2.6.17	USB D-Link DBT-120	128 MB	Pentium 2 265 MHz
Waysmall	Linux gumstix 2.6.18	UART Infineon Tech AG	64 MB	XScale-PXA255 400 MHz

Table 1: Bluetooth scanning sensors

scanning data, all Bluetooth sensors were configured to be in undiscoverable mode.

A description of the used sensors is outlined in Table 1.

As an experimental proof-of-concept we used a Gumstix Waysmall computer as one of the scanning devices [9]. A Waysmall is an extremely small computer which has a size smaller than a cigarette package. The model we used had a built-in Bluetooth device and support for memory extension by RS-MMC cards. Since this device features a very small size, it is easily hidden and can thus be deployed unobtrusively in many locations. The device can either be powered with batteries or an external power source. However, if a long term scan is intended, the external power source is preferable since the most common battery solutions only last for 4-6 hours. Since the manufacturer provides an open-source Linux distribution as the main OS option, it is easy to customize the functionality of the device. By preconfiguring and compiling a filesystem image, the deployment of multiple devices is easily done by transferring the preconfigured system image to the device and then plug it in at the intended location.

Bluetooth vulnerability testing on the attacker side was conducted using an IBM Thinkpad T43 laptop with a Linux kernel version of 2.6.17.4. The Bluetooth chip used is manufactured by Broadcom Corporation and is part of the standard configuration of the used laptop.

4.3 Setup

For the experiment setup, we placed each of the four Bluetooth sensors in room CSE 101 (reception desk), CSE 222 (coffee room), CSE 302 and CSE 410 (both offices). The Bluetooth sensors ran the monitoring script every 40 seconds to collect traces of other Bluetooth devices within the capturing range (Fig. 1). The sensors were left running from 11/28/06 to 12/04/06.

4.4 Vulnerability testing

To test the prevalence of recent vulnerabilities in the Bluetooth implementation of mobile phones, we tested select models of different manufacturers. All of them are quite popular, representing different classes of phones:

- Sony Ericsson T610 (2003): A very popular older phone, known to be vulnerable to *Blue-Snarfing*.
- Sony Ericsson K700i (2004): An older, feature-rich phone, successor to the T610 series.
- Nokia N70 (2005): A typical *SymbianOS* smartphone, rich of features and connection options.
- Motorola MOTORAZR V3 (2005): A very popular phone, currently in the medium to lower price range.
- Sony Ericsson W810i (2006): A newer multimedia phone.
- LG Chocolate KG800 (2006): A new, stylish multimedia phone.

Probing for vulnerabilities was done with the following tools or methods:

- **Bluetooth stack smasher**: BSS is a Bluetooth *fuzzer*, sending out random malformed packages at the lowest L2CAP layer of Bluetooth [12]. We used the currently available version 0.6 to test our candidates for vulnerabilities on this level with 11 different modes.
- **SonyEricsson display reset**: With BSS, an exploit in the SonyEricsson phones K600i, V600i, K750i and W800i was found in 2005: the phones reacted abnormal to a special L2CAP packet, turning the screen slowly off for about 45 seconds [13]. We used the proof-of-concept code to test if phones from that model series are still vulnerable.

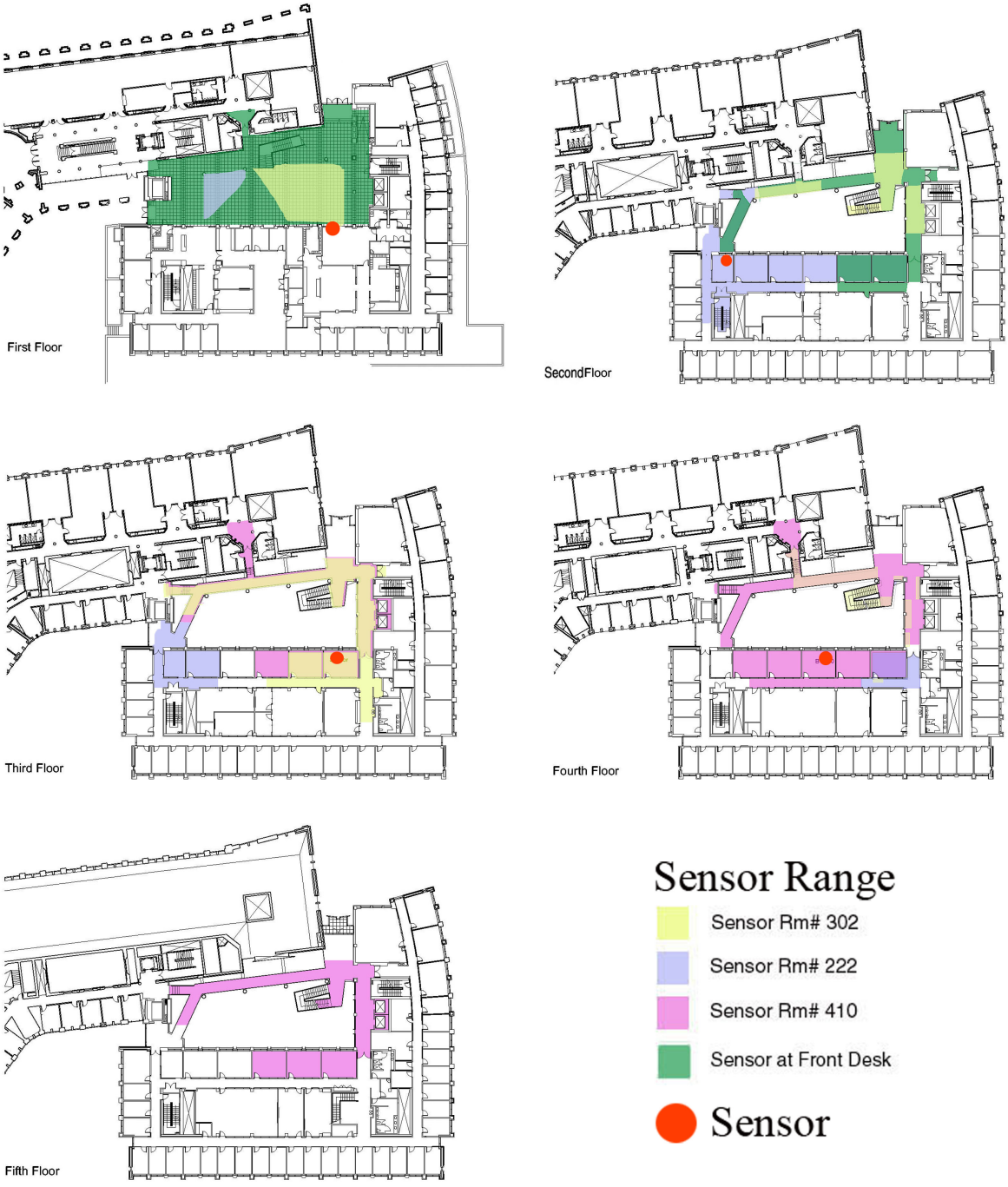

Figure 1: Floor plans with sensor ranges

- **OBEXpush DoS**: Our first approach to the DoS attack was to continuously try to open a connection to a device on a random RFCOMM channel. Using this technique, the targeted phone would continuously prompt the user to decide whether the attacking device would be allowed to access the corresponding service on the phone. However, the success of this technique was limited since it was too slow and some of the phones ignored the requests after a while. We then explored other options to try to keep a phone busy over Bluetooth and found that the OBEX Push protocol provided a more rigid and effective way of user interruption. By continuously trying to push a file through this protocol, the user was flooded with prompts whether to accept the file or not, which disabled any other usage of the phone, including the ability to turn off Bluetooth. Using this method also added the bonus that if the user ever were to accept the file, it could potentially contain malicious code which would compromise the device.

- **Bluediving**: This Bluetooth security tool provides a user interface to test a number of possible exploits on a target, including the aforementioned attacks *BlueSnarf* and *BlueBug* [16]. We used the current version 0.5 mainly to test for the targets for different versions of Bluesnarf vulnerabilities, also to confirm the known exploit on the SonyEricsson T610. Also, we tried to make use of the BlueBug vulnerability where possible.

- **Crafted vCards**: Some phones are known to react abnormal when handling specially crafted files in the vCard format[17]. An unexpected long name in the contact details of that file cause an overflow, crashing Bluetooth or the complete phone. We test the handling of vCards, by sending such an abnormal file, and make the phones open and store the contact details.

5 Analysis

This section provides some insights into the captured Bluetooth traffic behaviourally and device-demographically as well as giving an overview of the results of the vulnerability testing.

5.1 Traces

This section outlines what information we were able to collect about the behaviour of the subjects and their device statistics in a relatively short capture session. The range of the sensors used in this study was also relatively large, which brought up issues about the accuracy of an individual's location, particularly in relation to what floor of the CSE building they were on.

5.1.1 Bluetracking and Bluecoffee

Armed with data from a network of Bluetooth sensors, it becomes possible to characterize the behaviour of the passers-by over time. While our capture session generated more than 28,000 Bluetooth events and detected 160 unique devices, we restricted the set of targets to 15 known devices from the student of the class that were willing to participate. While more information about social networking, interaction and interpersonal behaviour could also be inferred based on this data, we were unable to pursue this direction due to time constraints on the analysis.

Several of the traced students generated on the order of 1000 or more events, which enables even the slowest of observers to fairly rapidly determine what their behaviour is like for given days of the week. Just one of the students we tracked generated more than 1750 events over the course of 6 days and it became fairly easy to tell where he or she would be at any particular time of the day. Even with a very limited set of data about an individual, it is still possible to determine their routine - when do they come to university? When do they leave? What time do they take lunch? Are there any exceptions? Does it change for weekends?

With models of behaviour in hand about specific individuals, an attacker can easily determine when and where target individuals for a variety of malicious behaviour including theft of personal belongings when the attacker knows that the target will be away from their valuables or for locating the target in order to accost them for marketing. Our traces enabled us to observe the regular routines of several of the subject students.

While the range and number of sensors provides some limitations on the accuracy of determining precisely where the target is, with a bit of common sense it is still fairly easy to figure out where people are and at what times. If the number of sensors is small with a broader range than the attacker might like for more precise results, the sensors can be positioned such that

the overlap of the sensors provides the extra layer of detail. If we assume that the attacker knows some more information about the subject, such as their office location for instance, then their specific location can be readily inferred. An undergraduate student that agreed to be tracked spent a large amount of time in particular places that overlapped with the sensors, which allowed our trace to identify his or her position very quickly.

An aspect of our study that we thought would provide some useful behavioural insight was the Bluetooth sensor in the CSE graduate coffee room. Our particular setting up made it easy to infer from the Bluetooth traces when and how often each individual come to get coffee, as well as the amount of time he or she spent in the coffee room. Despite the traces being captured during 'crunch time' for a lot of the grad students, we did not capture as many devices in the small hours of the morning, or more frequent trips to the coffee room as we had anticipated.

5.2 Vulnerabilities

In Table 2 the success of the previously mentioned attacks is described. We confirmed the serious bug in the SE T610 phones, even with a newer firmware version. Considering that this phone is no longer sold, it is unlikely to be an increasing problem. All other phones handled the Bluesnarf attack correctly: When not paired, connection was refused or not even answered. After a pairing every potentially harmful operation like a deletion had to be confirmed at the phone itself.

The attack by a crafted vCard file is also no longer possible. The phones we studied either refused opening the file ("File corrupted") or shortened the overlong name appropriately.

Only the Nokia N70 Smartphone was vulnerable to Bluetooth Stack Smasher attacks, in particular attack modes 2, 6, 8, 10 standing for L2CAP Connection Request, Disconnection Request, Echo Request and Info Request. After a successful attack, the phone is no longer able to use Bluetooth. Turning Bluetooth off and on again restores functionality.

A surprising result was that all studied phones were to some degree vulnerable to Denial-of-Service via OBEX object push. There is no way to defend against that attack, except turning Bluetooth off in advance. Once being attacked, the only solution is to walk out of range. Turning the phone off does not help, because once it is turned on again within range of the attacker, it immediately gets attacked again. Using only one Bluetooth dongle, it was possible to keep three phones busy at the same time. By using multiple dongles in parallel, even more targets can be effectively attacked.

Motorola also provide a feature called "Find me" which makes it impossible to leave the phone in discoverable mode. Instead of permanently setting the device to be discoverable, it was only possible to make it discoverable for a period of 60 seconds. This efficiently reduces the possibility to trace and attack the phone, since it is much harder to find.

Using Blueprinting we were able to get the unique identification of the phones we examined. With the available data at [4], we could confirm the Blueprint of the SonyEricsson T610, even having a different, newer firmware version. We will contribute the data to the database at `www.trifinite.org`, together with the vulnerabilities that we could or could not confirm.

6 Discussion

As our study indicates, there are still a number of issues with Bluetooth concerning privacy and security.

Visibility

Even though the Motorola V3 phone is one of the cheaper phones in our list, it is a good archetype for Bluetooth implementation on mobile phones. It is not vulnerable to most of our tested attacks, and its "Find Me" feature provides a basic protection against privacy intrusion. We see no reason why it would be beneficial to have a continuously visible device in normal everyday use. However, just because a device is in undiscoverable mode it does not guarantee that a bluetooth sensor will not be able to detect it. As mentioned in [15], it is possible to detect undiscoverable devices by doing a bruteforce search of the Bluetooth address domain. A device which is in range of a sensor can also be pinged using a feature of the L2CAP protocol. This may be exploited by constantly trying to ping previously discovered devices and thus effectively determine when they are in the vicinity.

OBEX DoS

Another possible issue worth noting is that the OBEX Push protocol does not require a previous pairing or authentication between devices. Anyone can try to push a file to a user's device, however the user must explicitly accept the file transfer. By combining an OBEX push with a prior message (which could be "sent" using *Bluejacking* [7]) an attacker could pretend to be a trusted authority (e.g. a service provider, phone

phone model	firmware version	vulnerabilities					
		BSS	OBEX DoS	display reset	Bluesnarf	Bluebug	vCard
SonyEriccsson T610	R6C005	No	No [1]	No	Yes	No	No
SonyEricsson K700i	R2A041	No	Yes	No	No	No	No
Nokia N70	5.0609.2.0.1	Yes	Partly [2]	No	No	No	No
Motorola MOTORAZR V3	R374_G_0E.42.10R_A	No	Partly [3]	No	No	No	No
SonyEriccsson W810i	R4DB005	No	Yes	No	No	No	No
LG Chocolate KG800	p64-esa-v10i	No [4]	Yes	No	No	No	No

[1] OBEX push is not accepted, but a DoS with RFCOMM connects was possible instead
[2] if Bluetooth functionality is set to a hotkey, it is sporadically possible to turn it off but the sent file has to be accepted
[3] occasionally a disconnect with the hang-up-key is able to end the attack
[4] however during the scanning the phone does hardly react to user input, making it impossible to use

Table 2: Vulnerabilities for different phone models

manufacturer, etc.) and instruct the user to install a "security patch". A potential future research direction could be to evaluate how many users would eventually accept a file which is continuously pushed to their device.

Using the DoS attack described in this paper, it would be possible to create an efficient Bluetooth jammer with very limited resources and little effort. For example, this could be done by deploying a script on a hidden gumstix which continuously searches for devices and floods them with requests. Consider the implications of combining Bluetooth jamming with an efficient, traditional computer worm.

To prevent any unwanted requests, a basic Bluetooth "firewall" could at least enable the blacklisting of device addresses. However, because it is possible to spoof Bluetooth addresses on some devices, this would provide only rudimentary protection.

Improving the implementation

Our Bluetooth trace study could potentially be extended by using the *Received Signal Strength Indication* (RSSI), part of the Bluetooth specification since version 1.1, for triangulation. That enables a higher degree of localization accuracy, thus further intruding on the user's privacy. Future implementations could also push data directly to a centralized store, even if that limits the kinds of devices for sensors, and possibly their location.

Blueprinting

By using the *Blueprinting* technique [4], a potential Bluetooth worm could spread quickly and efficiently by launching device specific attacks. The blueprint would be used to determine which attack to use, thus eliminating the need to test all possible attacks on every discovered device. This would potentially make the worm spread faster since more devices that are around for only a limited time could be infected with a higher probability.

Privacy

People who are sensitive to their privacy may feel uneasy about being traceable from a Bluetooth device. This kind of tracking can be avoided by turning Bluetooth off completely or at the very least off of discoverable to prevent initial capture of the device MAC address. Particularly in the case of Bluetooth-enabled mobile devices, the other method of avoiding Bluetracking would be to leave your phone behind, but we find this to be an unreasonable solution.

This study is intended to show how easy it is for observers to determine behavioural models of people using simple Bluetooth sensors that can be set up using a network of devices, or just adding some capturing processes to existing PCs. The latter is a more probably threat, since one can fairly easily attach a Bluetooth dongle and install capture scripts to run in the background. While this study had a limited geographical scope, it could easily be extended to be in Internet cafes, public libraries or university machines with reasonably open access.

With the prevalence of research and recent legislature to do with phishing attacks, *Bluechat* could also be employed as a potential phishing technique. While this study did not explore this due to the obvious privacy concerns, it is worth noting as a potential malicious use of Bluetooth.

7 Conclusions

We have studied the current vulnerabilities and privacy issues of the Bluetooth communication protocol. We have found that there are numerous mobile devices

which are in discoverable mode. During our scanning interval of one week, we found 160 unique devices which were left in discoverable mode. We have also found that many of the previously discovered Bluetooth vulnerabilities are no longer possible on newer phone models. However, it seems that most people still do not perceive Bluetooth tracking as a privacy threat, since they are not aware or do not care that their devices are left in discoverable mode. Detailed models of behaviour of people can still be found using a relatively easily deployable network of Bluetooth sensors. Once detected, the only way to prevent Bluetracking by this method is to turn Bluetooth off.

Furthermore, we found that flooding Bluetooth devices with OBEX Push request is a very simple but effective DoS attack, since phones usually force the user to decide whether to accept the incoming file or not by displaying pop-up notifications. Finally, we conclude that even though it is possible to trace devices by deploying a sensor network, detailed positioning would be easier with shorter range sensors.

Acknowledgements

We want to thank all the people who participated in our study, our professor Tadayoshi Kohno and the CSE reception staff for letting us put a gumstix on their window.

References

[1] Bluetooth Special Interest Group: The Bluetooth Technology Web Site
https://www.bluetooth.com/
Retrieved Dec 12th 2006

[2] Marc Haase and Jan Blumenthal: BlueTrack - Tracking Bluetooth devices, 2004
http://www-md.e-technik.uni-rostock.de/forschung/projekte/BlueTrack.htm
Retrieved Dec 2nd 2006

[3] AJ.Solon, MJ.Callaghan, J.Harkin and TM.McGinnity: Case Study on the Bluetooth Vulnerabilities in Mobile Devices, 2006
International Journal of Computer Science and Network Security, VOL.6 No.4, April 2006

[4] Martin Herfurt and Collin Mulliner.
Blueprinting - Remote Device Identification based on Bluetooth Fingerprinting Techniques, December 20, 2004
http://trifinite.org/Downloads/Blueprinting.pdf

[5] Jing Su, Kelvin K. W. Chan, Andrew G. Miklas, Kenneth Po, Ali Akhavan, Stefan Saroiu, Eyal de Lara, Ashvin Goel: A Preliminary Investigation of Worm Infections in a Bluetooth Environment, 2006
WORM'06, November 3, 2006

[6] The Shmoo Group: Braces - A Bluetooth Tracking Utility, 2004
http://braces.shmoo.com/
Retrieved December 11th 2006

[7] The Bunker: Bluetooth - Serious flaws in Bluetooth security lead to disclosure of personal data, 2004
http://www.thebunker.net/resources/bluetooth
Retrieved December 12th 2006

[8] Eamonn O'Neill, Vassilis Kostakos, Tim Kindberg, Ava Fatah gen. Schiek, Alan Penn, Danaë Stanton Fraser and Tim Jones: Instrumenting the city: developing methods for observing and understanding the digital cityscape, 2006
UbiComp 2006

[9] gumstix - way small computing
http://www.gumstix.com/
Retrieved November 30th 2006

[10] BlueZ - Official Linux Bluetooth protocol stack
http://www.bluez.org/
Retrieved December 11th 2006

[11] Spoofing the MAC address of a Gumstix
http://www.digitalmunition.com/TheftOfLinkKey.txt
Retrieved December 12th 2006

[12] Pierre Betouin, SecuObs.com: Bluetooth Stack Smasher, February 5th, 2006
http://www.secuobs.com/news/05022006-bluetooth10.shtml
Retrieved December 11th 2006

[13] Pierre Betouin, SecuObs.com: Display Reset Exploit, February 5th, 2006
http://www.secuobs.com/news/05022006-bluetooth7.shtml#english
Retrieved December 11th 2006

[14] IEEE Registration Authority - IEEE OUI and Company_id Assignments
http://standards.ieee.org/regauth/oui/
Retrieved November 16th 2006

[15] Ollie Whitehouse, atStake: *RedFang*
http://www.securiteam.com/tools/5JP0I1FAAE.html
Retrieved December 11th 2006

[16] Bluediving - Next generation Bluetooth security tool, June 15th, 2006
http://bluediving.sourceforge.net/
Retrieved December 9th 2006

[17] Marek Bialoglowy: Bluetooth Security Review Part 2, May 5th 2005
http://www.securityfocus.com/infocus/1836
Retrieved December 12th 2006