Studying Bluetooth Malware Propagation

The BlueBag Project

Current Bluetooth worms pose relatively little danger compared to Internet scanning worms—but things might change soon. The authors' BlueBag project shows targeted attacks through Bluetooth malware using proof-of-concept codes and devices that demonstrate their feasibility.

Luca Carettoni and Claudio Merloni
Secure Network Srl

Stefano Zanero
Politecnico di Milano

Thanks to its characteristics, Bluetooth is emerging as a pervasive technology that can support wireless communication in various contexts in everyday life. For this reason, it's important to understand the potential risks linked with various wireless devices and communication protocols. At present, the greatest level of diffusion exists in so-called smart phones. These devices offer all the functions of cutting-edge telephones while integrating those of advanced handheld computers managed by operating systems such as Symbian or Microsoft Windows Mobile.

Smart phones can send and receive SMSs, MMSs (multimedia messages), and email, plus let users listen to MP3 files, watch videos, surf the Internet, play games, manage agendas, synchronize and exchange data with their PCs, and much more. Although they still constitute a niche market, smart phones saw a growth rate of 100 percent per year for the past five years, and according to projections released at the beginning of 2006 by ABI Research, a market research company, they held 15 percent of the global cell phone market by the end of 2006, which is equivalent to 123 million units sold, thanks to growing user requests for applications such as mobile email, decreasing prices, and a broader choice of models (www.abiresearch.com/products/market_research/Smartphones).

Because smart phones are now very similar to PCs, they're simultaneously more vulnerable, more useful, and more attractive for potential attack than older mobile phones. This increased vulnerability is due to the presence of a system of evolved connectivity applications that expose the phone (and the data it contains) to risks. Fortunately, recent cell phone viruses haven't caused significant damage, except for the obvious inconveniences created when the phone malfunctions. This has led to the myth that Bluetooth malware is yet another form of viral code that doesn't pose any real or new security issues and which has a relatively low chance of causing significant damage. However, as we will show, the potential for the propagation of dangerous Bluetooth malware indeed exists. Until now, a combination of lucky chances and various environmental difficulties sheltered us from the widespread propagation of such epidemics, but we cannot simply keep crossing our fingers and hoping for the best.

In this article, we focus on the new risks created by the widespread presence of Bluetooth-enabled devices carrying both potentially sensitive data and vulnerability-prone software. In particular, we show how this mix of technologies could become a vehicle for propagating malware that's specifically crafted to extract information from smart phones. We built a mobile, covert attack device (which we call BlueBag) that demonstrates how stealthy attackers can reach and infect a wide number of devices.

Bluetooth technology

As a word or term, *Bluetooth* is now fairly common. The literal meaning supposedly refers to the Viking Emperor Harald (*Blåtand*, in Danish), who lived during the 10th century AD and united the kingdoms of Denmark, Norway, and Sweden (http://en.wikipedia.org/wiki/Harald_I_of_Denmark). In fact, the Bluetooth protocol aims to unify different wireless data-transmission technologies among mobile and static electronic devices such as PCs, cellular phones, notebooks, PDAs, DVD players, MP3 devices, TVs, Hi-Fis, cash registers, point-of-sale terminals, and even household appliances such as refrigerators and washing machines.

Bluetooth is essentially an alternative to the traditional infrared communication standards (the most famous being IrDA, Infrared Data Association). Whereas IrDA transmits data using infrared light waves, Bluetooth is based on short-wave radio technology, which can transmit data across physical obstacles without needing line of sight.[1] Bluetooth devices use the 2.4-GHz frequency range (the same range that WiFi 802.11 technology uses): the exact frequency spectrum used varies between countries due to national regulations.

Significant improvements over IrDA are the fact that it requires neither a line of sight nor proper orientation of devices; it offers the possibility of connecting multiple devices and not just pairs, as well as an increased range of connectivity. When individuals connect different Bluetooth devices together, they create personal area networks, or PANs (also called piconets in the Bluetooth specification), which are small ad hoc networks that can exchange data and information just as within regular LANs. These improvements are also the key reasons that Bluetooth can be used, for instance, to transport automatically spreading malware. This isn't true with IrDA because it requires proper alignments between both the transmitting and receiving devices, effectively avoiding "casual" or unwanted interaction.

Bluetooth technology is characterized by low power (from 1 to 100 milliwatts [mW]—a thousand times less than the transfer power of a GSM cell phone) and a communication speed of around 1 Mbit per second (Mbps). With regard to power, Bluetooth devices can be grouped in classes, each corresponding to a different reach:

- class 1 can communicate with Bluetooth devices in a 100 meters range;
- class 2 can communicate with Bluetooth devices up to a 10 m range; and
- class 3 can communicate with Bluetooth devices within a 1 m range.

Currently, most common devices belong to classes 2 and 3; laptops and cell phones, for instance, normally use class 2 peripherals. Toward the end of 2004, a new implementation of the Bluetooth technology (version 2.0) was released that allowed transfer speeds of up to 2 and 3 Mbps, as well as lower energy consumption. The new protocol is also backward-compatible.

Security issues

Although the Bluetooth standard incorporates very robust security mechanisms[2] that application developers can use to create secure architectures, researchers have discovered a series of theoretical glitches and possible attacks in Bluetooth's core specifications.[3,4] The most serious of these[5] can lead to a compromise of the cryptographic algorithm protecting communication through sniffing, but this attack is impractical because the attacker must be present at the pairing of devices and then must be able to sniff communications between them. This is more difficult than it seems: Bluetooth divides the 2.4-GHz spectrum range into 79 channels, through which devices hop in a pseudorandom sequence that differs from PAN to PAN. This is done both to avoid interference among different PANs and to enhance security. In fact, this inhibits common commercial Bluetooth hardware from sniffing communications in a PAN it doesn't participate in (contrast this with common, off-the-shelf WiFi hardware, which can be placed in monitor mode and used for sniffing). A hardware sniffer can easily cost in the range of US$10,000, which places this attack out of reach for the common aggressor, but surely within the reach of corporate spies. Provided to be able to sniff the pairing, a tool exists for personal identification number (PIN) cracking (www.nruns.com/security_tools.php). As a possible solution, researchers have proposed alternate implementations of Bluetooth with more secure encryption algorithms.[6]

Specific attacks

Even if Bluetooth is theoretically quite robust, several security issues have surfaced in various implementations of the standard stack since late 2003. Among the existing attacks, we can quote significant examples drawn from www.trifinite.org, an organization that hosts information and research in wireless communications:

- *BlueSnarf*. This type of attack uses the OBEX (object exchange) Push service, which is commonly used to exchange files such as business cards. BlueSnarf allows an attacker to access the vulnerable device's phone book and calendar without authentication. A recently upgraded version of this attack gives the attacker full read–write access.
- *Bluejacking*. By carefully crafting the identification that devices exchange during association, attackers can transmit short deceitful text messages into authentication dialogs. Users can then be tricked into using their access codes, thereby authorizing an aggressor to access a phone book, calendar, or file residing on the device.
- *BlueBug*. This vulnerability permits access to the cell phone's set of "AT commands," which let an aggressor use the phone's services, including placing outgoing calls, sending, receiving, or deleting SMSs, diverting calls, and so on.
- *BlueBump*. This attack takes advantage of a weakness in the handling of Bluetooth link keys, giving devices that are no longer authorized the ability to access services as if still paired. It can lead to data theft or to the abuse of mobile Internet connectivity services, such as Wireless

Application Protocol (WAP) and General Packet Radio Services (GPRS).

- *BlueSmack*. This denial-of-service (DoS) attack knocks out certain types of devices; attackers can perform it with standard tools.
- *HeloMoto*. A combination of BlueSnarf and BlueBug, this attack's name comes from the fact that it was originally discovered on Motorola phones.
- *BlueDump*. This attack causes a Bluetooth device to dump its stored link key, creating an opportunity for key-exchange sniffing or for another pairing to occur with the attacker's device of choice.
- *CarWhisperer*. This attack abuses the default configuration of many hands-free and headset devices, which come with fixed PINs for pairing and transmission.
- *BlueChop*. This DoS attack can disrupt any established Bluetooth piconet by means of a device that isn't participating in it, if the piconet master supports multiple connections.

These flaws demonstrate how, in many cases, attackers can steal information from mobile devices, control them from a distance, make calls, send messages, or connect to the Internet. In computer systems, these problems are traditionally resolved with the release and application of patches. That same approach doesn't extend to GSM handsets, however; in most cases, a firmware update can be performed only at service points and shops, not by users. Therefore, many phones and firmwares can be vulnerable and in use long after a vulnerability is discovered and a patch produced.

Some of these attacks are implemented in Blooover, a proof-of-concept application that runs on Symbian cell phones (www.trifinite.org/trifinite_stuff_blooover.html). This counters the idea that attackers need laptops to execute their attacks, therefore making themselves visible. Most of these attacks can also be performed at a distance using long-range antennae and modified Bluetooth dongles; a Bluetooth class-2 device was reportedly able to perform a BlueSnarf attack at an astounding distance of 1.08 miles (www.trifinite.org/trifinite_stuff_lds.html).

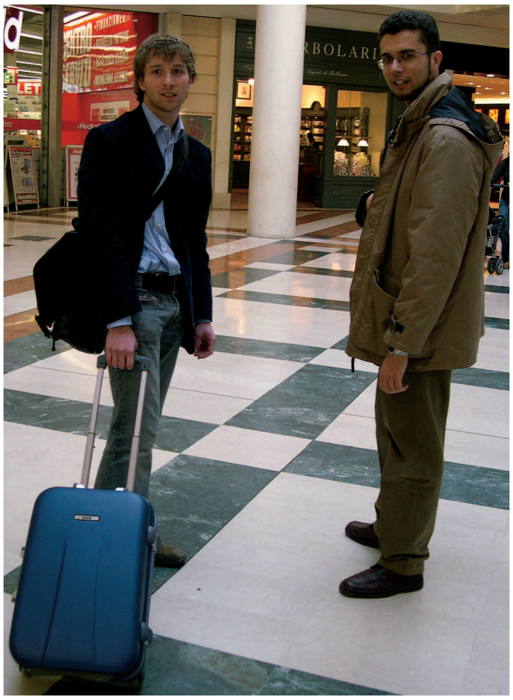


Figure 1. Luca Carettoni (left) and Stefano Zanero (right) with the BlueBag trolley. The picture was taken during the survey at the Orio Center shopping mall. Notice how inconspicuous the trolley is in this context, particularly if you keep in mind that the mall is in front of an airport.

Creating the BlueBag: A covert attack and scanning device

Our goals in undertaking this survey were to gather data on the prevalence of insecure devices to understand how susceptible people are to simple social engineering attacks, and to demonstrate the feasibility of attacks in secured areas such as airports or office buildings.

To mount any type of attack without being noticed, we needed to create a covert attack and scanning device, which we later came to call the *BlueBag* (see Figure 1).

We envisioned a Linux-based embedded system with several Bluetooth dongles to process many discovered devices in parallel, using an omnidirectional antenna to improve the range and cover a wide area. We needed both a hidden tool and an instrument that could easily be carried around and still have a long battery life.

To fulfill these requirements, we created the BlueBag by modifying a standard blue trolley and inserting a Mini-ITX system (see Figure 2) with the following off-the-shelf components:

- a VIA EPIA Mini-ITX motherboard (model PD6000E; because it doesn't have a fan, its power consumption is reduced);.
- 256 MBytes of RAM in a DDR400 DIMM module;
- EPIA MII PCI backplate to extend the available onboard USB connections from two to six;
- a 20-Gbyte iPod, with a 1.8-inch hard drive that can resist an acceleration of up to 3gs;
- eight class-1 Bluetooth dongles with Broadcom chipsets (some were connected to a four-port USB hub);
- a modified class-1 Linksys Bluetooth dongle (Cambridge Silicon Radio chipset) modified with a Netgear omnidirectional antenna with 5dBi gain.
- a picoPSU, DC-DC converter (this small power supply can generate up to 120 watts at over 96 percent efficiency); and
- a 12V-26Ah lead acid battery to power our lengthy surveying sessions (up to 8 hours).

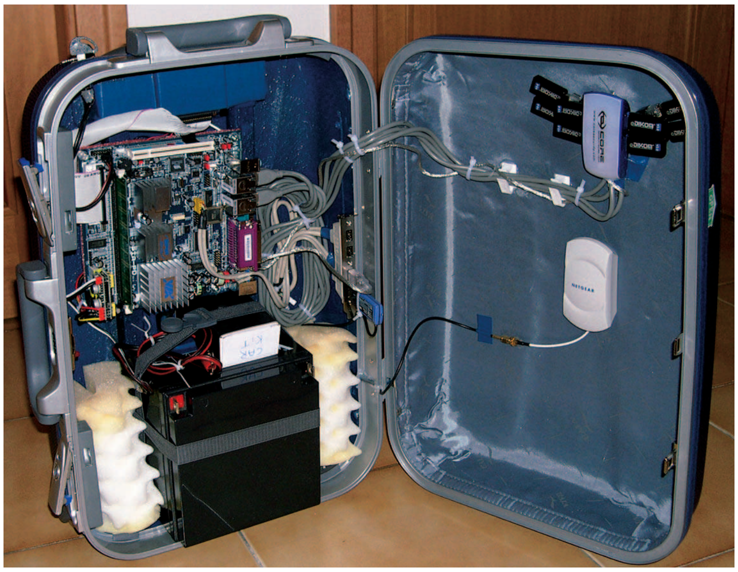


Figure 2. The BlueBag open. Note the motherboard (top, left side) and battery (bottom, left side) as well as the dongles (top, right side) and the antenna (below the dongles).

The total cost to build such a device is approximately US$750, demonstrating just how economical it is to create a Bluetooth attack device.

The BlueBag runs on GNU/Linux OS (specifically, we use the Gentoo distribution for its outstanding customizability and performance), on top of which we created a software infrastructure in Python that makes it easy to devise, control, and perform survey sessions. The software is completely multithreaded, and we can use the available dongles to perform different tasks concurrently. We implemented a simple but useful dongle management and allocation scheme to dynamically learn about available resources and lock them when needed. By doing so, we can reserve specific dongles to run applications that need to lock single physical interfaces for some time (the "pand" daemon, which allows us to establish connectivity over Bluetooth). The software is quite modular and was designed with the typical producer/consumer pattern: producers put found devices in a queue, using the standard utilities that come with BlueZ (the official Linux Bluetooth stack) in order to collect information. The software also includes customized versions of well-known Bluetooth information-gathering techniques such as *blueprinting* (a method for remotely identifying Bluetooth-enabled devices, similar to OS fingerprinting). A distinct thread manages the queue and assigns tasks to different consumers.

We designed the BlueBag software suite to allow us to monitor and control the test's execution from a palmtop or smart phone via a Web interface that runs on top of a TCP/IP over Bluetooth connection. Using this configuration, there's no need to open the BlueBag case in public. At no time did anyone stop us or suspect us of doing something unusual, even in highly secured areas such as airports.

Survey results: A discomforting landscape

In our surveys, we initially focused on identifying how many active Bluetooth devices were in discoverable (or visible) mode. This is, in fact, the condition of potential real-world risks: researchers have demonstrated that it's possible to find devices with active Bluetooth technology in nondiscoverable mode using a brute-force attack. However, given the enormous time expenditure this would entail, it isn't feasible in a generic context. An attack with this method is possible only if attackers want to target a specific device they know to be active and in range, and even then, they must first identify the brand and model in order to prune the address space.

Therefore, keeping a phone in nondiscoverable mode provides a basic form of protection against targeted attacks, and, in general, keeps the device safe from worms that use Bluetooth technology to replicate, given that such worms research their victims by simply scanning devices in the area. For this reason, our test focused exclusively on detecting devices in discoverable mode—the only ones actually in a condition of potential risk of attack from Bluetooth malware.

We conducted our survey in several high-transit locations surrounding Milan:

- Milan's Exhibition Centre, during the InfoSecurity 2006 trade show;
- the Orio Center Shopping Mall;
- the MM2 Cadorna Metro Station;
- the Assago MilanoFiori Office District;
- Milan's Central Station;
- the Milan Malpensa Airport; and
- Politecnico di Milano Technical University, Leonardo Branch.

We chose a variety of venues to better evaluate whether and how the prevalence of potentially vulnerable targets varied in different contexts populated by different people. Milan's Central Station, for instance, has a very heterogeneous user base (and a dense crowd—the station serves 270,000 passengers on an average business day); the Orio Center Shopping Mall on a Saturday is filled with many young people and families, subjects who might not be aware of the dangers linked with new technologies, as opposed to visitors and exhibitors at the InfoSecurity trade show (which sees roughly 2,000 security professionals a day).

Table 1. Summary of surveying results.

LOCATION	DATE	DURATION (HH: MM)	UNIQUE DEVICES	DEVICE RATE
InfoSecurity 2006	02/08–10/06	4:42	149	0.53
Orio Center Shopping Mall	03/01–11/06	6:45	377	0.93
MM2 Metro Station	03/09/06	0:39	56	1.44
Assago Office District	03/09/06	2:27	236	1.60
Milan Central Station	03/09/06	1:12	185	2.57
Milan Malpensa Airport	03/13/06	4:25	321	1.21
Politecnico di Milano				
Technical University	03/14/06	2:48	81	0.48
Total		22:58	1405	

We performed multiple sessions, on different days, for a total of 23 hours of scanning dispersed over seven days. Table 1 shows the results; "unique devices" denotes the number of unique devices in discoverable mode that we found during a specific session, and "device rate" indicates the average number of unique devices discovered per minute.

This data shows the capillary diffusion of Bluetooth technology in everyday life and also highlights the huge number of potentially vulnerable devices we found, even in such a short duration: at first glance, Bluetooth seems to be an integral part of everyone's life, important not only for professional but also for personal use. Note, too, that in terms of risk awareness among the Central Station, the Milan Malpensa Airport (populated by a heterogeneous public), and the Assago Office District (where most users use these devices for work purposes), there's an insignificant difference. The situation was significantly better—indicating a greater awareness among users—at the InfoSecurity conference and at the university.

Categorizing devices

For the 1,405 unique devices detected, we performed further analysis to broadly categorize the devices: cell and smart phones (1,312), PCs/notebooks (39), Palm Pilots (21), GPS navigators (15), printers (5), and other various devices (13). In a similar, independent experiment that FSecure performed in parallel during CeBIT 2006 (the ICT trade show in Hannover, Germany), a regular laptop device capable of identifying active Bluetooth devices in a 100-meter range found more than 12,500 devices with discoverable Bluetooth mode during a week of scanning (www.f-secure.com/weblog/archives/archive-032006.html). To our knowledge, the researchers made no attempt to break down the data any further.

After grouping the devices, we also tried analyzing the types of services the devices offered and, in particular, those that can be used to propagate worms. As Table 2 shows, the OBEX Push service was active and in range for enough time to allow the scanning of 313 devices; this service is normally used for transferring information (business cards, for instance) or files and applications—including worms. It's very likely that most, if not all, cell phones have the OBEX Push service activated. Because we found 1,312 phones among the devices, the result might seem strange at first sight. The explanation is simple: among all those devices, 313 stayed in range long enough to allow the OBEX Push service to let BlueBag correctly poll them.

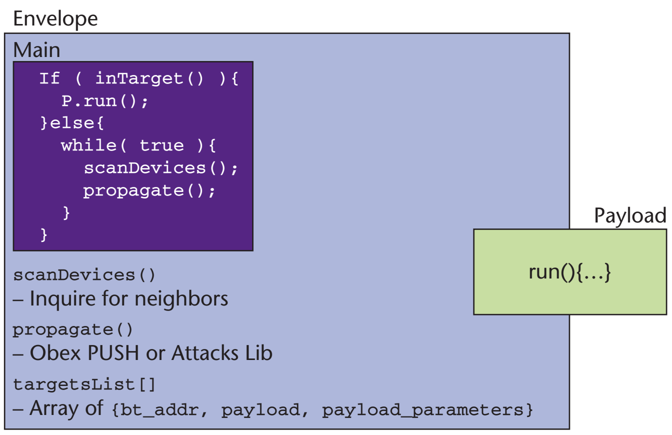
Table 2. Services offered by mobile devices.

SERVICE TYPE	NUMBER OF DEVICES
OBEX Object Push, OBEX file transfer	313
Headset hands-free audio gateway	303
Dial-up networking	292

Visibility

Another important finding from our survey was "visibility time"—that is, the average time in which a device remains in a potential attacker's range, or the time in which an aggressor could exploit the device. This time depends substantially on the different activity patterns of people in different contexts: for instance, at the Orio Center Shopping Mall, the average time was 12.3 seconds, at the Politecnico di Milano Technical University, 10.1, and in the Milano Malpensa Airport, the time was 23.1 seconds. Of course, in some cases, this time depends on the activity pattern a hypothetical aggressor might carry out: at the Politecnico, we deliberately avoided staying in a single classroom for a long time, but an aggressor interested in a specific target might very well do so, or he or she might follow the target in an airport up to the gate (where most people settle down to wait for boarding), thus extending this time. Our estimated average visibility times are therefore interesting for casual contacts, such as the one implied by casual worm transmission.

It's important to point out that some cell phone mod-



```
If ( inTarget() ){
  P.run();
}else{
  while( true ){
    scanDevices();
    propagate();
  }
}
```

`scanDevices()`
– Inquire for neighbors

`propagate()`
– Obex PUSH or Attacks Lib

`targetsList[]`
– Array of `{bt_addr, payload, payload_parameters}`

Figure 3. Pseudocode of a Bluetooth worm with dynamic payloads for targeted attacks.

els on the market are configured to be in discoverable mode by default if the Bluetooth connection is activated, thus requiring the user to manually modify the setting to the secure, nondiscoverable mode. Other devices must instead be manually brought to discoverable mode and are automatically reset to nondiscoverable after a short time period. Our survey showed this to be effective: just a handful of the detected device models were of the latter type, surely out of proportion with the respective market shares. Because keeping devices in nondiscoverable mode doesn't prevent communication among paired devices, keeping a phone in nondiscoverable mode shouldn't entail a heavy usability burden.

Social engineering

After we investigated how effectively Bluetooth malware can propagate, we realized that we needed to also estimate the success rate of the basic social engineering techniques Bluetooth worms commonly use. Most existing worms rely on the user accepting a file to propagate, so we wanted to know the ratio of users who would accept an unknown file transfer from an unknown source. To obtain this data, we developed an OBEX Pusher, an add-on to our normal survey scripts, which searches for all discoverable Bluetooth devices with OBEX Push support enabled and then sends them a file. Using this tool (and transmitting an innocuous image file), we found that an astounding 7.5 percent of device owners carelessly accepted unknown file transfers from unknown sources and were thus highly vulnerable to social engineering attacks.

Bluetooth-enabled malware networks

Our experiments show that just a small percentage of people today are aware of the risks incurred by using apparently innocuous devices. Moreover, smart phones and connected palmtops have become daily work tools for people with medium to high levels of responsibility within their organizations. This implies that these devices could hold particularly interesting information that potential aggressors might want, such as for industrial espionage.

All the elements are thus in place for a huge risk, to both companies and individuals; we can almost certainly foresee an increase in attacks that aim not only to make a mobile device unusable or connect it to premium-rate telephone numbers but also target specific information on the device.

The effort it takes to reach a target device is often thought of as a form of protection. To prove this assumption wrong, we created a network of viral agents that can spread among mobile devices looking for a target, zero in on it, and then report information back to the attacker.

Because such agents are targeted to a specific environment or person, it's interesting to study the use of dynamic payloads that vary depending on the type of infected device. We designed a proof-of-concept worm infrastructure that uses an envelope-payload mechanism (see Figure 3).

The envelope component is a piece of software that can scan for Bluetooth devices and propagate to found devices; it has a list of targets to propagate to and a set of payloads that it can "deploy" on the targets. The payload components can be any type of malicious code that we want to execute on victim devices within the limits of cell phone operating systems—examples include keyloggers, audio recorders, and sniffers. A similar design pattern (in a very different context) appears in the Metasploit framework's Meterpreter infrastructure.[7]

Such payloads can also use the high connectivity of Bluetooth-enabled devices to transmit harvested information back to the attacker (in much the same way that common PC-based spyware does), for instance, using the Internet email service or a sequence of MMSs. In this way, the attacked device doesn't need to be within the attacker's range to send the retrieved data. It's not difficult then to envision an attacker that infects several devices (during a morning commute, for example) belonging to an organization's employees, and then just waits for one of these devices to reach and infect or attack the device of the organization's CEO. In other words, attackers could create a botnet of Bluetooth-enabled, remotely controlled zombie machines, which they could then use to perform further attacks on devices they couldn't normally reach.

One of the barriers to mobile malware propagation has historically been differences among various operating systems and hardware platforms. This is becoming easier to overcome because of the growing popularity of Java 2 Micro Edition (J2ME), which enables software authors (and, correspondingly, malware authors) to create cross-platform software for mobiles. We successfully implemented our proof of concept in Java; it runs on any cell phone compatible with Mobile Information Device Profile (MIDP) 2.0 on which JSR-82 (the Java Bluetooth API) is active.

Features that would make this worm really dangerous (and that we therefore didn't implement) are ways to auto-execute with as little interaction with the device user as possible. On Symbian phones, for instance, a worm can overwrite system files due to various structural flaws in access control. Otherwise, implementation flaws and bugs that allow for command execution (such as the ones we described earlier) could help this worm propagate.

Simulation results

To correctly evaluate the threat this attack scenario poses, we developed a model and a simulation to understand its effectiveness. Due to space limitations, we refer the reader to other work[8,9] for a full discussion of the problems involved with modeling computer virus propagation. An excellent analysis of mathematics for infectious diseases in the biological world is available elsewhere.[10]

Traditional propagation models

Propagation models evolve naturally, following the changes in viruses' propagation vectors. The earliest models targeted virus propagation through the infection of host executables.[11] Most biological epidemiological models share two assumptions: they're homogeneous—that is, an infected individual is equally likely to infect any other individual—and they're symmetric, which means there's no privileged direction of virus transmission. The former makes these models inappropriate for illnesses that require noncasual contact for transmission, as well as being inappropriate for describing the early stages of propagation of an epidemic that's strongly location-dependent. In an influential seminal paper, Jeffrey Kephart and Steve White addressed these shortcomings by transferring a biological model onto a directed random graph to better approximate the chain of software distribution and the way it worked in the early days of the personal computing revolution.[11]

Among other results, Kephart and White showed that the more sparse a graph is, the more slowly an infection on it spreads; there's also a higher probability that an epidemic condition doesn't occur. (In a *sparse* graph, each node has a small, constant average degree; in a *local* graph, the probability of having a vertex between nodes B and C is significantly higher if both have a vertex connected to the same node A.)

Mass mailers and scanning worms

The introduction of the Internet changed the malware landscape and made traditional models unrealistic. The first effect was the appearance of mass-mailing worms, which demonstrated that tricking users into executing the worm code attached to an email or exploiting a vulnerability in a common email client to automatically launch it were successful ways to propagate viral code. One of the best models for such propagation occurs when the email service is modeled as an undirected graph of relationships between people.[12] The problems here lie in how to model the users' behavior,[13] that is, what to do if the worm doesn't automatically exploit a vulnerability but instead relies on social engineering, and how to build the relationship graph (which is more a *local* than a *sparse* one).

Eugene Spafford wrote the first description of self-propagating worms that scan for vulnerabilities.[14] In recent years, such worms have changed the threat landscape once more. They can be modeled through the random constant spread (RCS) model,[15] developed using empirical data derived from the outbreak of the Code Red worm, a typical random scanning worm. This model uses extremely rough approximations, ignoring the effect of immunization and recovery. It implicitly assumes that the worm will peak before a remedy begins to be deployed. Additionally, it models the Internet as an undirected, completely connected graph. This is far from true,[16] but the model still behaves macroscopically well. UDP-based worms, however, require corrections to account for bandwidth restrictions and bottleneck Internet links.[17]

Bluetooth virus propagation can happen in several different ways, but the most common until now has been through simple social engineering. The worm sends messages with copies of itself to any device in range through an OBEX Push connection. The receiver, finding a seemingly innocuous message on the cell phone with an invitation to download and install an unknown program, often has no clue that it can pose a danger. Cabir, one of the first cell phone worms and the first case of malware that could replicate itself solely through Bluetooth, used this technique.

MMS messages are another potential medium of propagation. The Commwarrior worm propagated through MMS (in fact, it spread from 8 a.m. to midnight using Bluetooth connections and from midnight to 7 a.m. through MMS messages). Another method of propagation would be the use of email- or TCP-based worms, such as the ones usually seen on PCs, although such methods haven't really been used in phone viruses until recently.

By the end of May 2006, F-Secure research laboratories had classified more than 140 virus specimen (www.f-secure.com/v-descs/mobile-description-index.shtml). Of these, most found in the wild propagate by relying solely on Bluetooth technology. In fact, our own experiments showed that this transmission method alone can reach 7.5 percent of a mixed population of targets, so we decided to simulate the propagation of viral code that uses Bluetooth as its vector.

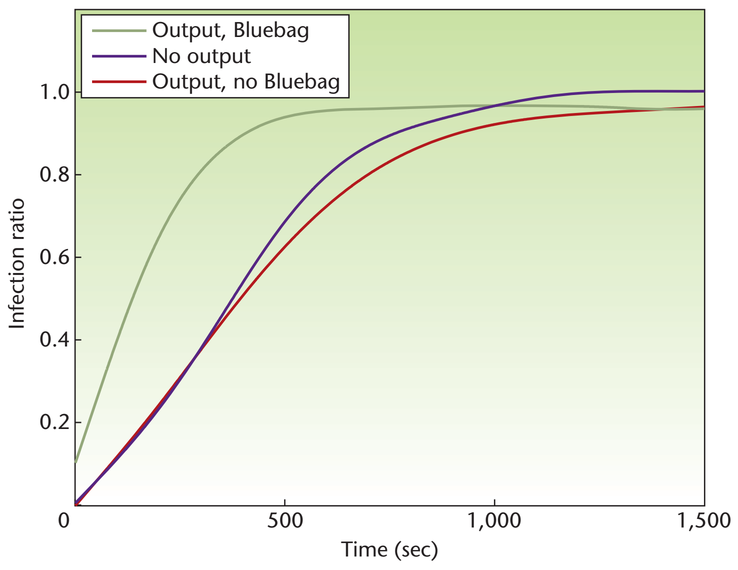


Figure 4. Infection ratio. Our simulation examined three different conditions: without people entering or leaving the area, with a flow of people and a propagating worm, and with a flow of people and the BlueBag actively disseminating a worm.

This wasn't an easy task. On one hand, we wanted to follow the early stages of a worm's propagation because we wanted to evaluate its effectiveness as a targeted attack tool instead of as a global infection (thus assumptions of homogeneity and nonlocality can't hold). On the other hand, we needed to simulate the occasional and volatile interactions of highly mobile devices. So we needed to effectively simulate a highly sparse graph of relations that change dramatically over time.

We used results from the ad hoc network research community to simulate the transient geographical relationships caused by the movement of people in physical places.[18] Cecelia Mascolo and Mirco Musolesi's CMM-Tool generates realistic traces of movement for people and their respective devices. We developed a small simulator that takes such feed as an input and then reproduces the behavior of a Bluetooth worm that propagates across them. The resulting BlueSim tool can replicate, under various hypotheses, the behavior of real worm propagation, taking into account the visibility time of the devices, the inquire time needed, the data transfer rate, and so on. We chose not to analyze layer-1 radio aspects such as collisions and interference problems, which could potentially occur in crowded places with many devices; to do so, we would have needed a complete network simulator such as NS, which in turn would have required a lot more computational power to complete even simple simulations.[19]

To evaluate how effectively a targeted worm can propagate through a population, we recreated different specific contexts with fixed parameters inspired from real environment characteristics and data collected during our survey. In particular, we simulated a shopping mall—a simplified version of the Orio Center Shopping Mall we visited—with 250 × 100 meters of surface and 78 shops. We considered a population of 184 discoverable devices (7.5 percent of which were susceptible to infection), with a Bluetooth transmission range of 15 meters, which is reasonable for mobile phones or PDAs. We conservatively estimated a 0.3-Mbps bandwidth link and a 42-Kbyte worm—the size of the envelope-and-payload worm we designed.

In our first scenario, we used CMMTool to mimic the behavior of people inside lunch areas or food courts, creating groups of relatively stationary people, a small number of whom "travel" among lunch areas. Figure 4 shows the results. Initially, we didn't consider people entering or leaving the shopping mall during our simulation time (on the line marked "no output"). We then added a random flow of people with discoverable devices entering and exiting the mall (on average, one person each 10 seconds, a realistic value from our assessments). We then tested two different conditions: the first was a worm propagating (starting with just one infected device), marked "no BlueBag" in the figure, and the second was the presence of an attacker with a tool similar to our BlueBag, who was actively disseminating a worm.

As Figure 4 shows, after little more than 30 minutes on average (the time of a typical lunch break), a simple worm could infect any susceptible device in the lunch area through propagation alone. An attacker with a device such as the BlueBag would obtain the result even faster.

In a second scenario, we considered the behavior of a more mobile crowd of people walking in and out of shops and browsing the window displays. In this case, the results were similar, but they depended heavily on motion patterns in the mall and were slower than in the food court scenario (propagation speed was nearly halved in this case).

In this work, we tried to envision possible future attack scenarios involving targeted malware propagated through Bluetooth-enabled covert attack devices. We demonstrated the existence of a very high risk potential, created by low awareness, ever-increasing functionalities and complexity, and by the feasibility of targeted, covert attacks through Bluetooth-enabled malware.

Possible future extensions of this work include better planning of the malware's "phone home" payload, to understand how likely it is for the collected data to reach the attacker under various scenarios and how to improve worm auto execution and process hiding. The creation of a Bluetooth-only command and control infrastructure would be a challenging evolution because it would integrate ad hoc networking issues in our work.

Like common worms, our malware doesn't currently use Bluetooth attacks to spread itself: in the future, we want to investigate whether we can use a sort of attack library, combining social engineering attacks and Bluetooth technology attacks.

Another possible extension would be the use of BlueBag as a honeypot, to "capture" Bluetooth worms in the wild and measure their real prevalence. We briefly engaged in this activity, but more extensive testing is needed to give reasonable statistical results. ▢

Acknowledgments

An earlier version of this work was presented at the Black Hat conference (www.blackhat.com). We thank Jeff Moss and the Black Hat staff for their support. F-Secure, an antivirus vendor based in Helsinki, Finland, and Secure Network, a security consulting and training firm based in Milano, Italy, jointly funded the early stages of the BlueBag project. One of the authors had partial support from the Italian Ministry of University and Research under the FIRB-PERF Project, in the research unit led by Giuseppe Serazzi, whose support we gratefully acknowledge. We warmly thank Martin Herfurt, Marcel Holtman, and Adam Laurie, authors of the earliest works on Bluetooth security issues, for their comments on this work. We also thank Mirco Musolesi (UCL) and Paolo Costa (DEI - Politecnico di Milano) for their help with modeling movement. Several people helped with various stages of this project, including Alvise Biffi, Laura Mantovani, Miska Reppo, and Mara Rottigni. Finally, we thank the anonymous reviewers for their helpful and extensive reviews of the first draft of this article.

References

1. R. Morrow, *Bluetooth Implementation and Use*, McGraw-Hill Professional, 2002.
2. C. Gehrmann, J. Persson, and B. Smeets, *Bluetooth Security*, Artech House, 2004.
3. M. Jakobsson and S. Wetzel, "Security Weaknesses in Bluetooth," *Proc. 2001 Conf. Topics Cryptology* (CT-RSA 01), Springer-Verlag, 2001, pp. 176–191.
4. S.F. Hager and C.T. Midkiff, "Demonstrating Vulnerabilities in Bluetooth Security," *Proc. IEEE Global Telecommunications Conf.* (GLOBECOM 03), vol. 3, 2003, IEEE CS Press, pp. 1420–1424.
5. Y. Shaked and A. Wool, "Cracking the Bluetooth Pin," *Proc. 3rd Int'l Conf. Mobile Systems, Applications, and Services* (MobiSys 05), ACM Press, 2005, pp. 39–50.
6. P. Hamalainen et al., "Design and Implementation of an Enhanced Security Layer for Bluetooth," *Proc. 8th Int'l Conf. Telecommunications* (ConTEL 2005), vol. 2, 2005, IEEE CS Press, pp. 575–582.
7. K.K. Mookhey and P. Singh, "Metasploit Framework"; www.securityfocus.com/infocus/1789, July 2004.
8. S.R. White, "Open Problems in Computer Virus Research," *Proc. Virus Bulletin Conf.*, 1998.
9. E. Filiol, M. Helenius, and S. Zanero, "Open Problems in Computer Virology," *J. Computer Virology*, vol. 1, nos. 3–4, 2006, pp. 55–66.
10. H.W. Hethcote, "The Mathematics of Infectious Diseases," *SIAM Rev.*, vol. 42, no. 4, 2000, pp. 599–653.
11. J.O. Kephart and S.R. White, "Directed-Graph Epidemiological Models of Computer Viruses," *Proc. IEEE Symp. Security and Privacy*, 1991, IEEE CS Press, pp. 343–361.
12. C.C. Zou, D. Towsley, and W. Gong, *Email Virus Propagation Modeling and Analysis*, tech report, TR-CSE-03-04, Univ. of Massachusetts, Amherst, 2003.
13. S. Zanero, "Issues in Modeling User Behavior in Computer Virus Propagation," *Proc. 1st Int'l Workshop on the Theory of Computer Viruses*, 2006.
14. E.H. Spafford, "Crisis and Aftermath," *Comm. ACM*, vol. 32, no. 6, ACM Press, 1989, pp. 678–687.
15. S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time," *Proc. 11th Usenix Security Symp.*, (Security 02), Usenix Assoc., 2002, pp. 149–167.
16. A. Ahuja, C. Labovitz, and M. Bailey, *Shining Light on Dark Address Space*, tech. report, Arbor Networks, Nov. 2001; www.arbornetworks.com/downloads/research38/dark_address_space.pdf.
17. G. Serazzi and S. Zanero, "Computer Virus Propagation Models," M.C. Calzarossa and E. Gelenbe, eds, *Tutorials 11th IEEE/ACM Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecommunications Systems* (MASCOTS 2003), Springer-Verlag, 2003.
18. M. Musolesi and C. Mascolo, "A Community-Based Mobility Model for Ad Hoc Network Research," *Proc. 2nd ACM/SIGMOBILE Int'l Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality* (REALMAN 06), ACM Press, 2006, pp. 31–38.
19. C.-J. Hsu and Y.-J. Joung, "An Ns-Based Bluetooth Topology Construction Simulation Environment," *Proc. 36th Ann. Symp. Simulation* (ANSS 03), 2003, IEEE CS Press, p. 145.

***Claudio Merloni** is a senior consultant for Secure Network S.r.l., an information security company based in Milan, Italy. His research interests are auditing, policy development, and risk assessment activities, particularly in a banking environment. Claudio holds a MSc degree in computer engineering from the Politecnico di Milano university. Contact him at c.merloni@securenetwork.it.*

***Luca Carettoni** is a senior consultant for Secure Network S.r.l., an information security company based in Milan, Italy. His research interests are in Web application security. A regular contributor of OWASP-Italy, he has led penetration testing efforts on several Italian and European banks. Luca holds a MSc degree in computer engineering from the Politecnico di Milano university. Contact him at l.carettoni@securenetwork.it.*

***Stefano Zanero** holds a PhD in computer engineering from the Politecnico of Milano university, where he is currently spending a post-doc period. His research interests include the development of intrusion detection systems based on unsupervised learning algorithms, Web application security, and computer virology. He is a member of the board of Journal in Computer Virology. Zanero is a member of the IEEE and the ACM, and a founding member of the Italian chapter of Information Systems Security Association (ISSA). Contact him at stefano.zanero@polimi.it.*